**ЛАБОРАТОРНАЯ РАБОТА № 4**

**СОЗДАНИЕ ЭЛЕМЕНТОВ ИНФОГРАФИКИ**

***Цель работы*** – создание приложений, которые наглядно представляют с помощью графика или диаграммы набор числовых данных, заданных таблицей.

## ТЕОРИЯ И ПОЯСНЕНИЯ

***Инфографика*** — это изображение, передающее смысл, данные, информацию с помощью графики, не текста. [<http://infographer.ru/vsyo-taki-chto-zhe-takoe-infografika/>].

### *Программирование изображений в среде Lazarus*

Изображение можно создать в любом компоненте, имеющем свойство Canvas.

**Canvas**, как объект класса TCanvas (мольберт со всеми инструментами и красками), обладает свойствами Pen, Brush и Font:

**Pen**, как объект класса TPen (карандаш для рисования контуров), имеет свойства Color (перечислимый тип TColor) и Style (перечислимый тип TPenStyle);

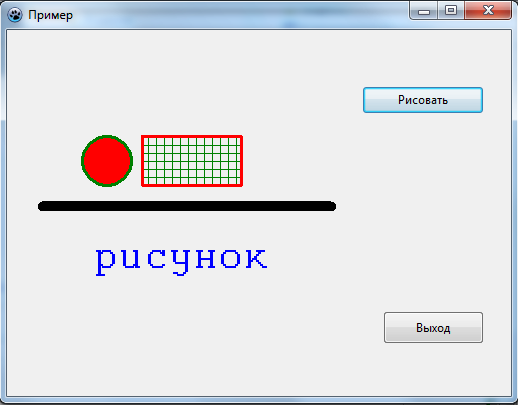
**Brush**, как объект класса TBrush (кисть для заливки внутри контура), имеет свойства Color и BrushStyle (перечислимый тип TBrushStyle);

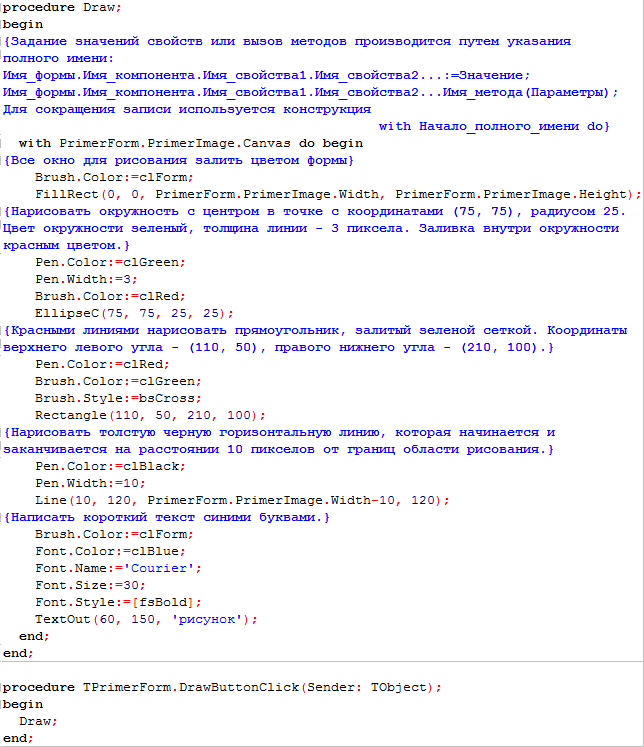
**Font**, как объект класса TFont (шрифт), имеет свойства Color, Size (тип Integer) и FontStyle (перечислимый тип TFontStyle).

Описания типов, имена и значения констант приведены в файле ..\Lazarus\lcl\graphics.pp

Изображение создается как набор графических примитивов: точка, линия, полилиния, прямоугольник, эллипс и т.п. Рисование примитивов и вывод текста выполняют методы класса TCanvas, реализованные в виде процедур и функций.

### *Пример программирования изображения.*



******

### *Точечное изображение графика*

Пусть задана функция , и требуется построить ее график в области значений аргумента []. Представим график в виде последовательности точек. Для этого нужно задать *n* значений аргумента , вычислить значения функции в этих точках и нарисовать точки . Очевидно, что функция на этом интервале должна быть непрерывной, в противном случае график не отобразит истинную зависимость.

Координаты  могут быть заданы заранее или могут рассчитываться, исходя из их равномерного расположения на оси.

### *Кусочно-линейное представление графика*

Пусть задана функция , и требуется построить ее график в области значений аргумента []. Представим график в виде последовательности отрезков прямых, т.е. в виде ломаной линии. Для этого нужно задать *n* значений аргумента , вычислить значения функции в этих точках и нарисовать отрезки между точками  и . Очевидно, что функция на этом интервале должна быть непрерывной, в противном случае график не отобразит истинную зависимость.

Координаты  могут быть заданы заранее или могут рассчитываться, исходя из их равномерного расположения на оси.

### *Физическая и экранная декартовы системы координат*

Физическая декартова система координат задается точкой начала координат и двумя осями: осью абсцисс (ось значений аргумента *x*), осью ординат (ось значений функции *y*). Экранная система координат тоже декартова и задается горизонтальной и вертикальной осями, начало координат совпадает с верхним левым углом окна компонента, на котором размещается изображение. Отсчеты по осям ведутся пикселями, т.е. точками изображения. Координаты четырех углов окна компонента с изображением можно записать следующим образом:

левый верхний – (0, 0);

левый нижний – (0, имя\_компонента.Height);

правый верхний – (имя\_компонента.Width, 0);

правый нижний – (имя\_компонента.Width, имя\_компонента.Height).

Для того чтобы график на экране выглядел так же, как график, построенный в физической системе координат (например, на миллиметровой бумаге), нужно выполнить преобразование физических координат в экранные, используя правила подобия.

Пусть в физической системе координат задана область значений аргумента . Тогда область возможных значений функции  – минимальное и максимальное значения, которых достигает функция в указанной области значений аргумента.

Зададим координаты области окна компонента, в которой нужно изобразить график: левый верхний угол – (), правый нижний угол – ().

Выберем точку с физическими координатами (), отображаемую в точку с экранными координатами ().

()

()

()

()

()

()

|  |  |
| --- | --- |
| Физическая система координат | Экранная система координат |

Запишем правила подобия расстояний между соответствующими точками в физической и экранной системах координат. Выведем соотношения, на основе которых физические координаты следует пересчитывать в экранные.

 ; (I.5.1)

 . (I.5.2)

В модуле, где строится график, координаты границ областей удобно задать в виде переменных или констант, описанных в разделе **implementation**, а формулы для расчета экранных координат оформить как функции с одним параметром,  или  соответственно. Поскольку экранные координаты должны быть целочисленными, следует выполнить округление результата. В дальнейшем, при построении графика, рисовании и подписывании осей, масштабировании или сдвиге области значений аргумента или функции, координаты любой экранной точки вычисляются с помощью этих функций.

### *Рисование столбчатой диаграммы*

Рисование столбчатой диаграммы выполняется с использованием преобразований из физической системы координат в экранную. Заметим, что по горизонтальной оси будет размещаться определенное количество столбцов. Это позволяет рассчитать ширину столбца напрямую в пикселах, т.е. преобразование координат из физической системы в экранную для горизонтальной оси делать не нужно. Высоту столбцов следует вычислять, пользуясь соотношениями (I.5.2) для вертикальной оси, оси функции *y*.

Рисование отдельного столбца выполняет метод **FillRect( )**.

### *Физическая полярная и экранная декартова системы координат*

В физической полярной системе координат начало координат *O* называется полюсом. Луч, исходящий из полюса, называется полярной осью. Местоположение любой точки *T* на плоскости задают две величины:

1) угол **α** между полярной осью и прямой, проходящей через полюс и точку;

2) расстояние ***r*** от полюса до точки.

*Y*

*ym*

*r*

*T*

*α*

*xm*

*O*

*X*

[<http://www.academiaxxi.ru/WWW_Books/HM/Ag/02/01/t.htm>]

Декартовы координаты точки выражаются через ее полярные координаты соотношениями:

Набор правил расчета экранных декартовых координат по физическим полярным координатам зависит от решаемой задачи. Следует определить, какими методами графического объекта можно нарисовать требуемое изображение, чтобы знать типы параметров процедур рисования.

### *Рисование круговой диаграммы*

Круговая диаграмма представляет собой набор отдельных секторов, которые в сумме образуют круг. Общий размер диаграммы задается радиусом круга, размер каждого сектора – углом, который пропорционален значению показателя, соответствующего этому сектору.

Рисование отдельного сектора выполняет метод **Pie( )**.

Procedure Pie (X1 , Y1 , X2 , Y2 , X3 , Y3 , X4 , Y4 : Integer ) ;

Метод Pie рисует замкнутую фигуру - сектор окружности или эллипса с помощью текущих параметров пера Pen, внутренняя область закрашивается текущей кистью Brush. Точки (X1,Y1) и (X2,Y2) задают прямоугольник, описывающий эллипс. Начальная точка дуги определяется пересечением эллипса с прямой, проходящей через его центр и точку (X3,Y3). Конечная точка дуги определяется пересечением эллипса с прямой, проходящей через его центр и точку (X4,Y4). Дуга рисуется против часовой стрелки от начальной до конечной точки. Рисуются прямые, ограничивающие сектор и проходящие через центр эллипса и точки (X3,Y3) и (X4,Y4).

[<https://teacher.ucoz.net/Lection/Pascal/glava_10.pdf>]

При рисовании круговой диаграммы достаточно знать углы отдельных секторов. Местоположение и размер диаграммы на форме вычисляются на основе размеров графического объекта и координат одной точки, например, верхнего левого угла квадрата, описывающего круг.

## ЗАДАНИЯ

### ИНСТРУМЕНТЫ ДЛЯ РИСОВАНИЯ

Разработайте приложение, которое создает изображение, полностью совпадающее с «Примером программирования изображения».

### ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ

Разработайте приложение, которое рисует точечное изображение или кусочно-линейное представление графика непрерывной однозначной функции, заданной аналитически, т.е. формулой. Функцию произвольно выбирает разработчик.

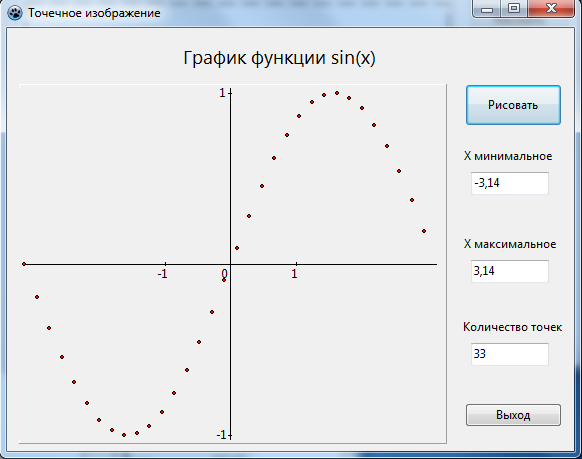
На форме должны присутствовать следующие элементы управления:

- окна ввода минимального и максимального значения аргумента;

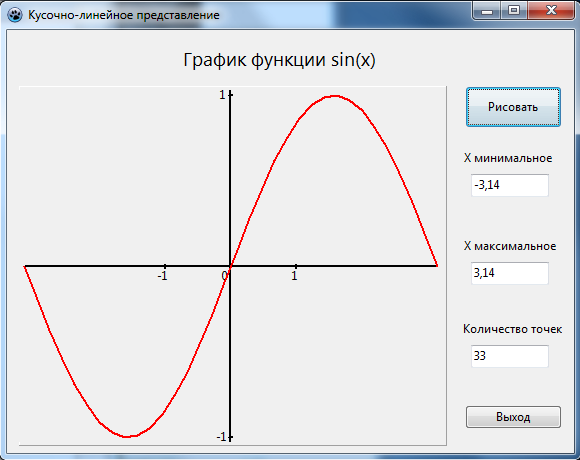
- окно ввода количества точек на интервале от минимального до максимального значения аргумента, в которых вычисляется значение функции. Предполагается, что эти точки (значения аргумента) равноудалены друг от друга.

**Замечание**. Пересчет физических координат в экранные координаты ДОЛЖЕН выполняться отдельными функциями, как это описано в I.5.

Возможный вид формы с точечным изображением графика:



Возможный вид формы с кусочно-линейным представлением графика:

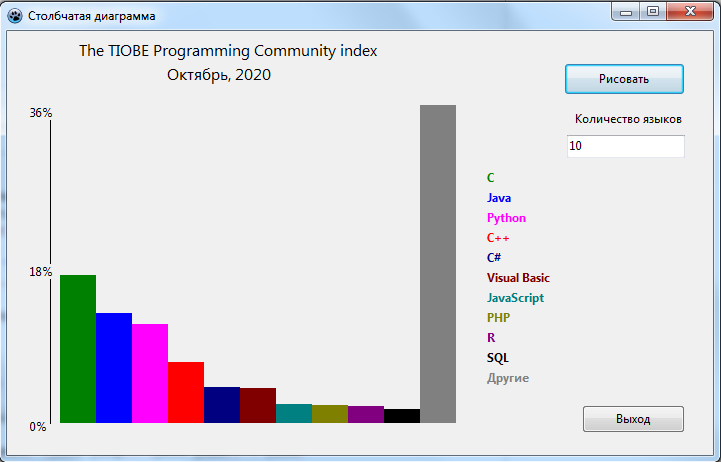


**Бонусное задание**. На форме для рисования графика замените окно количества точек на окно выбора количества точек из заранее заполненного списка (класс TComboBox).

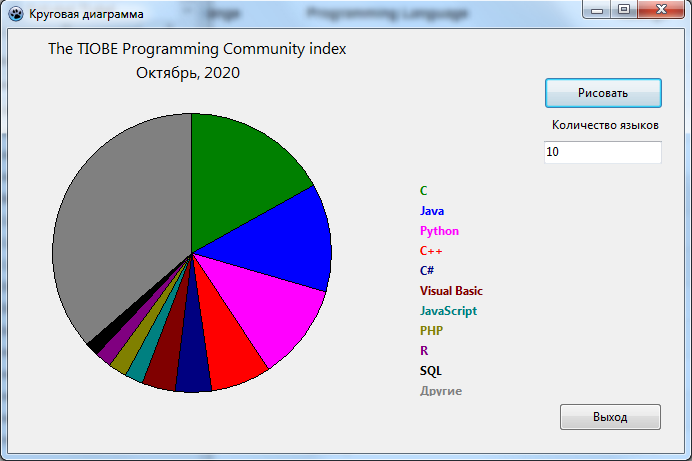
### ПОСТРОЕНИЕ ДИАГРАММЫ

Разработайте приложение, которое создает столбчатую или круговую диаграмму рейтинга нескольких языков программирования по результатам 2020 года из документа <https://www.tiobe.com/tiobe-index/>. Количество языков – от 5 до 10. Предусмотрите столбец «Другие языки», чтобы сумма рейтингов получилась равной 100%.

Возможный вид формы со столбчатой диаграммой:

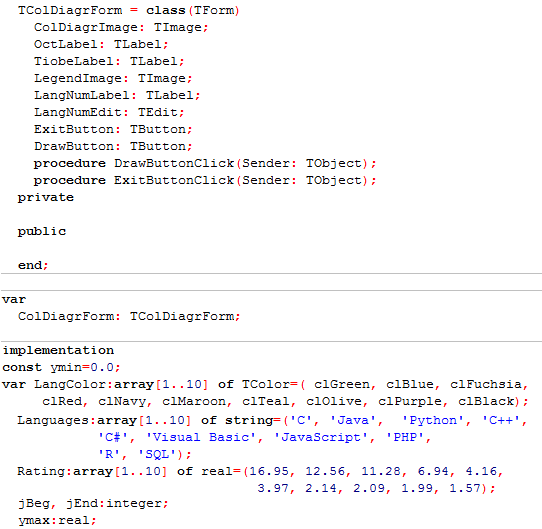


Возможный вид формы с круговой диаграммой:

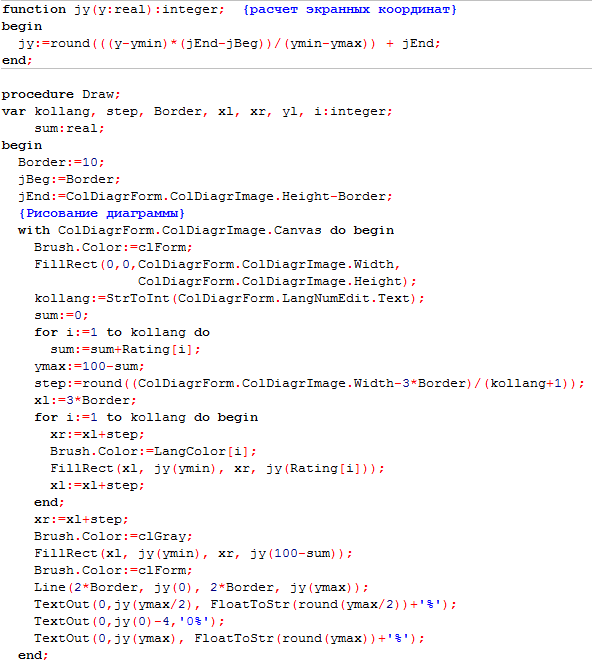


**Фрагменты программного кода модуля рисования столбчатой диаграммы**

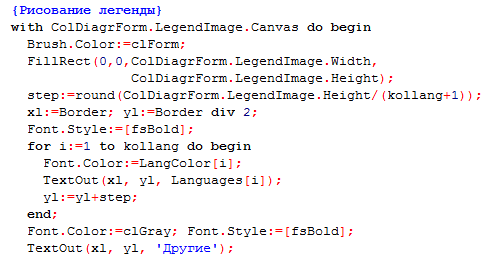
Форма и исходные данные



Рисование диаграммы

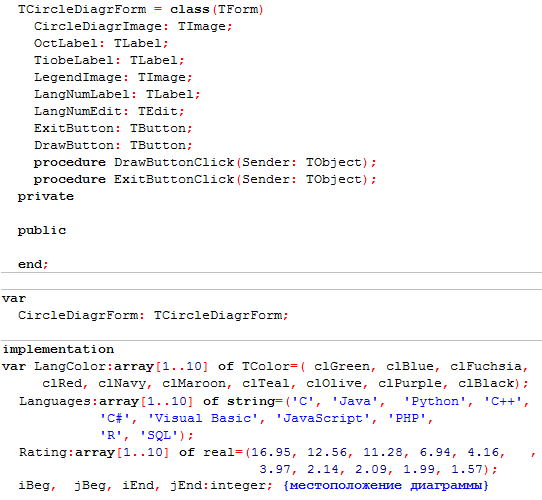


Рисование легенды

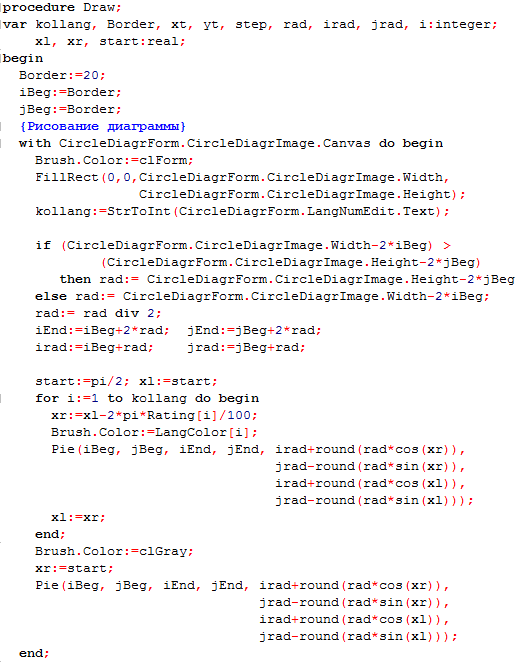


**Фрагменты программного кода модуля рисования круговой диаграммы**

Форма и исходные данные



Рисование диаграммы



Рисование легенды

