ПОСТРОЕНИЕ ГРАФИКОВ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ СРЕДСТВАМИ СРЕДЫ ПРОГРАММИРОВАНИЯ BORLAND DELPHI 7

Ваничкина А.Р.

МБОУ «Дединовская школа-интернат среднего общего образования», 9 класс

Руководитель: Холодных А.А., МБОУ «Дединовская школа-интернат среднего общего образования», учитель математики

Данная статья является реферативным изложением основной работы. Полный текст научной работы, приложения, иллюстрации и иные дополнительные материалы доступны на сайте VI Международного конкурса научно-исследовательских и творческих работ учащихся «Старт в науке» по ссылке: https://school-science.ru/6/7/38330.

Тема моей исследовательской работы «Построение графиков элементарных функций средствами среды программирования Borland Delphi 7».

Данная научно-исследовательская работа является актуальной в силу четырех обстоятельств:

Она имеет непосредственное практическое отношение к курсу информатики в школе;

Элементы работы могут быть представлены для изучения учащимся 9–11 классов;

Возможность применения разработанного приложения для проверки знаний учащихся, связанных построением графиков элементарных функций;

Работа имеет прикладной характер и требует углубленного изучения языка программирования Pascal в объектно-ориентированной среде программирования Borland Delphi 7.

Цель научно-исследовательской работы: разработать приложение для построения графиков элементарных функций средствами среды программирования Borland Delphi 7.

Построение графика функции лежит в основе исследования элементарных функций в курсе математики средней школы. График функции используют для решения задач и уравнений графическим способом. Задачи, связанные с графиками элементарных функций, внесены в программу единого государственного экзамена по многим предметам.

В связи с этим, разработка данного приложения позволит учащимся средней и, в особенности, старшей школы более плодотворно подготовиться к экзаменам и освоить свойства элементарных функций, изучаемых в школе.

Для достижения цели были поставлены следующие задачи:

1. Изучить процесс исследования элементарных функций в курсе средней школы;
2. Изучить функционал среды программирования Borland Delphi 7 и ее возможности;
3. Изучить основы программирования на языке Pascal;
4. Составить приложение для построения графиков элементарных функций средствами среды программирования Borland Delphi 7.

Научно-исследовательская работа состоит из: введения, двух глав, заключения и списка литературы.

В введение сформулированы цель, задачи и актуальность научно-исследовательской работы.

В первой главе рассмотрен функционал среды программирования Borland Delphi 7 и ее возможности для построения графиков.

Во второй главе рассмотрен процесс разработки приложения для построения графиков элементарных функций средствами среды программирования Borland Delphi 7.

В заключении отражены достигнутые результаты и перспективы развития приложения.

1. Среда программирования Borland Delphi 7 для построения графиков элементарных функций

Материал данной главы взят из учебника «Программирование в Delphi 7» [1].

Delphi – одна из самых мощных систем, позволяющих на самом современном уровне создавать как отдельные прикладные программы Windows, так и разветвленные комплексы, предназначенные для работы в корпоративных сетях и в Интернет.

Компонент Image и некоторые его свойства

Нередко возникает потребность украсить свое приложение какими-то изображениями. Это может быть графическая заставка, являющаяся логотипом вашего приложения. Или это могут быть фотографии сотрудников некого учреждения при разработке приложения, работающего с базой данных этого учреждения. В первом случае вам потребуется компонент Image, расположенный на странице Additional би-

блиотеки компонентов, во втором – его аналог DBImage, связанный с данными и расположенный на странице Data Controls.

Если установить свойство AutoSize в true, то размер компонента Image будет автоматически подгоняться под размер помещенной в него картинки. Если же свойство AutoSize установлено в false, то изображение может не поместиться в компонент или, наоборот, площадь компонента может оказаться много больше площади изображения.

Другое свойство – Stretch позволяет подгонять не компонент под размер рисунка, а рисунок под размер компонента. Установите AutoSize в false, растяните или сожмите размер компонента Image и установите Stretch в true. Вы увидите, что рисунок займет всю площадь компонента, но поскольку вряд ли реально вручную установить размеры Image точно пропорциональными размеру рисунка, то изображение исказится. Устанавливать Stretch в true может иметь смысл только для каких-то узоров, но не для картинок. Свойство Stretch не действует на изображения пиктограмм, которые не могут изменять своих размеров.

Свойство – Center, установленное в true, центрирует изображение на площади Image, если размер компонента больше размера рисунка.

Рассмотрим еще одно свойство – Transparent (прозрачность). Если Transparent равно true, то изображение в Image становится прозрачным. Это можно использовать для наложения изображений друг на друга. Одно из возможных применений этого свойства – наложение на картинку надписей, выполненных в виде битовой матрицы. Эти надписи можно сделать с помощью встроенной в Delphi программы Image Editor.

Канва и пикселы

Многие компоненты в Delphi имеют свойство Canvas (канва, холст), представляющее собой область компонента, на которой можно рисовать или отображать готовые изображения. Это свойство имеют формы, графические компоненты Image, PaintBox, BitMap и многие другие. Канва содержит свойства и методы, существенно упрощающие графику Delphi. Все сложные взаимодействия с системой спрятаны для пользователя, так что рисовать в Delphi может человек, совершенно не искушенный в машинной графике.

Каждая точка канвы имеет координаты X и Y. Система координат канвы, как и везде в Delphi, имеет началом левый верхний угол канвы. Координата X возрастает при перемещении слева направо, а координата Y – при перемещении сверху вниз. С координатами вы уже имели дело многократно, но пока вас не очень интересовало, что стоит за ними, в каких единицах они измеряются. Координаты измеряются в пикселах. Пиксел – это наименьший элемент поверхности рисунка, с которым можно манипулировать. Важнейшее свойство пиксела – его цвет. Для описания цвета используется тип TColor. С цветом вы встречаетесь практически в каждом компоненте и знаете, что в Delphi определено множество констант типа TColor. Одни из них непосредственно определяют цвета (например clBlue – синий), другие определяют цвета элементов окон, которые могут меняться в зависимости от выбранной пользователем палитры цветов Windows (например, ulBtnFace – цвет поверхности кнопок).

Но для графики иногда этих предопределенных констант иногда не хватает. Вам могут понадобиться такие оттенки, которых нет в стандартных палитрах. В этом случае можно задавать цвет 4–байтовым шестнадцатеричным числом, три младших разряда которого представляют собой интенсивности синего, зеленого и красного цвета соответственно.

Например, значение SOOFFOOQO соответствует чистому синему цвету, SOOOOFFOO – чистому зеленому, SOOOOOOFF – чистому красному. $00000000 – черный цвет, SOOFFFFFF – белый.

Рисование с помощью пера Pen

У канвы имеется свойство Pen – перо. Это объект, в свою очередь имеющий ряд свойств. Одно из них уже известное вам свойство Color – цвет, которым наносится рисунок. Второе свойство – Width (ширина линии). Ширина задается в пикселах. По умолчанию ширина равна 1. Свойство Style определяет вид линии. Это свойство может принимать следующие значения:

ps Solid	сплошная линия
psDash	штриховая линия
psDot	пунктирная линия
ps Dash Dot	штрихпунктирная линия
psDash DotDot	линия, чередующая штрих и два пунктира
ps Clear	отсутствие линии
pslnside Frame	сплошная линия, но при Width > 1 допускающая цвета, от личные от палитры Windows

Все стили со штрихами и пунктирами доступны только при Width = 1. В противном случае линии этих стилей рисуются как сплошные.

Стиль psInsideFrame – единственный, который допускает произвольные цвета. Цвет линии при остальных стилях округляется до ближайшего из палитры Windows.

У канвы имеется свойство PenPos типа TPoint. Это свойство определяет в координатах канвы текущую позицию пера. Перемещение пера без прорисовки линии, т.е. изменение PenPos, производится методом канвы MoveTo(X.Y). Здесь X и Y – координаты точки, в которую перемещается перо. Эта текущая точка становится исходной, от которой методом LineTo(X,Y) можно провести линию в точку с координатами (X,Y). При этом текущая точка перемещается в конечную точку линии, и новый вызов LineTo будет проводить линию из этой новой текущей точки.

2. Разработка приложения для построения графиков элементарных функций, изучаемых в средней школе

Для построения графиков элементарных функций, изучаемых в средней школе будем использовать привило составления таблицы значений. К этим функциям относятся функции:

$$y = kx + b;$$

$$y = ax^2 + bx + c;$$

$$y = \frac{k}{x};$$

$$y = \sqrt[n]{x};$$

В настоящий момент разработанное приложение позволяет производить построение графиков функций 1 и 2. Это обусловлено тем, что построение графиков данных функций зависит только от коэффициентов этих функций и сводится к вычислению значения этих функций при различных значениях аргумента *x*.

Таким образом, для построения графиков линейных функций необходимо знать коэффициенты *k* и *b*, а для построения графиков квадратичных функций соответственно *a*, *b* и *c*.

В связи с вышесказанным, входными параметрами будут выступать коэффициенты и отрезок, на котором будет производиться построение графиков данных функций. Для ввода всех необходимых данных будем использовать объект TEdit, который позволяет вводить с клавиатуры значения и присваивать им соответствующие переменные. Для вывода построенного графика нам необходим объект TImage, а для запуска программы будем использовать объект Tbutton (кнопка), построение графиков будем производить при помощи канвы рисунка (параметр Canvas). Также добавим кнопку очистки интерфейса программы и выхода из программы. Тогда весь интерфейс приложения будет состоять из 7 объектов TEdit, 3 кнопок и объекта TImage, в котором будет отображаться полученный результат (рис. 1).

Значения, которые записаны в полях TEdit, имеют строковый тип, для перевода их в числовой тип будем использовать встроенную функцию StrToInt, которая переводит значение строки в число.

```
k:=StrToInt(edt1.text);
b1:=StrToInt(edt2.text);
a:=StrToInt(edt3.text);
b2:=StrToInt(edt4.text);
c:=StrToInt(edt5.text);
```

Рис. 1

Для построения графиков функций необходимо найти значения функций при изменении значения аргумента x, который может принимать значения на отрезке $[x_{min}; x_{max}]$. Значения x_{min} и x_{max} задаются соответственно в объектах edt6 и edt7 Приращение аргумента положим равным 0,01. Тогда для построения графиков линейных функций код программы будет иметь следующий вид:

```
img1.Canvas.Pen.Color:=clBlue;
img1.Canvas.Pen.Width:=1;
y:=-k*x-b1;
img1.Canvas.MoveTo(Trunc(x*mush1+x),
Trunc(y*mush2)+y0);
Repeat
y:=-k*x-b1;
img1.Canvas.LineTo(Trunc(x*mush1+x0),
Trunc(y*mush2)+y0);
x:=x+0.01;
until x>=StrToInt(edt7.text);
```

Начало системы координат в Borland Delphi 7 расположено в верхнем правом углу, а сама система координат ориентирована так, что положительное направление оси абсцисс идет слева направо, а оси ординат сверху вниз. Поэтому для построения графика функции нам необходимо изменить знаки в общем виде линейной функции, то есть отразить ее относительно оси абсцисс. Первая и вторая строки в данном коде задают цвет графика и толщину в пикселях. Четвертая строка рассчитывает координаты для начала построения графика функции от минимального значения x. Функция Trunc откидывает дробную часть и оставляет только целую. Это позволяет построить график функции в объекте Timage, так как координаты в данном объекте могут быть только целыми числами. Цикл Repeat позволяет посчитать значения функции при каждом значении аргумента на заданном отрезке с шагом h=0,01. Функция LineTo чертит линии покоординатно, то есть строит линию от точки $\left(x; y\left(x\right)\right)$ до точки $\left(x+h; y\left(x+h\right)\right)$.

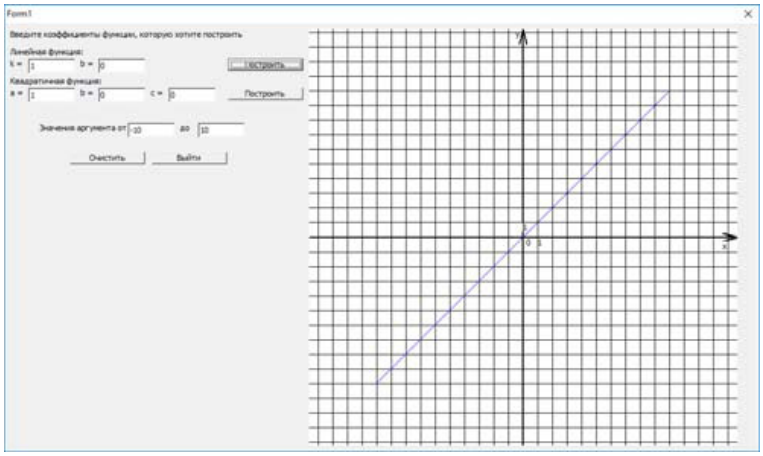

Рис. 2

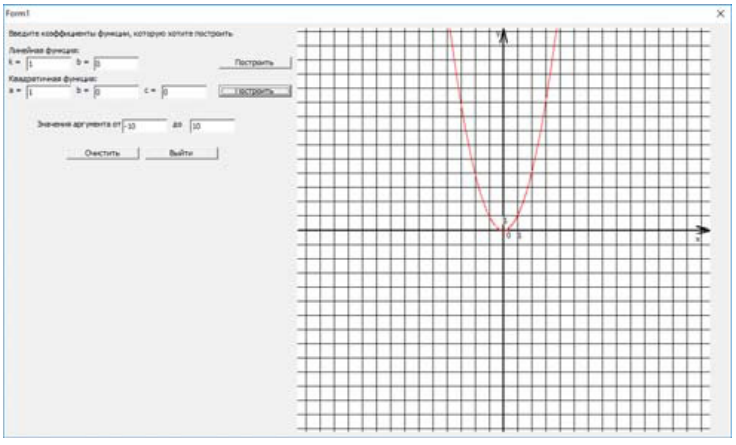

Рис. 3

Аналогичным образом происходит построение графиков квадратичных функций:

```
img1.Canvas.Pen.Color:=clRed;
img1.Canvas.Pen.Width:=1;
y:=-a*sqr(x)+b2*x+c;
img1.Canvas.MoveTo(Trunc(x*mush1+x),
Trunc(y*mush2));
Repeat
y:=-a*sqr(x)-b2*x-c;
img1.Canvas.LineTo(Trunc(x*mush1+x0),
Trunc(y*mush2)+y0);
x:=x+0.01;
until x>=StrToInt(edt7.text);
```

При запуске программы на построение графика линейной или квадратичной функции получим результат, изображенный на рис. *2, 3*.

Константы mush1 и mush2 используются для того, чтобы увеличить размера построенного графика. Это необходимо, чтобы график был читабельным, ведь объект Timage состоит из пикселей, расстояние между которыми очень маленькое, а значит если не применить масштабирование графика, то его просто не будет видно.

К тому же для комфортного изображения графиков данных функций необходимо построить координатные оси и координатную сетку. Для этого произведем дополнительные построения:

```
img1.Canvas.Pen.Color:=clBlack;
img1.Canvas.Pen.Width:=2;
img1.Canvas.MoveTo(x0,0);
img1.Canvas.LineTo(x0,ClientHeight);
img1.Canvas.MoveTo(0,y0);
img1.Canvas.LineTo(ClientWidth,y0);
img1.Canvas.MoveTo(x0,0);
img1.Canvas.LineTo(x0+(mush1 div
4),mush2);
img1.Canvas.MoveTo(x0,0);
img1.Canvas.LineTo(x0–(mush1 div
4),mush2);
img1.Canvas.MoveTo(ClientHeight,y0);
img1.Canvas.LineTo(ClientHeight-
mush1,y0–(mush2 div 4));
img1.Canvas.MoveTo(ClientHeight,y0);
img1.Canvas.LineTo(ClientHeight-
mush1,y0+(mush2 div 4));
img1.Canvas.TextOut(x0+mush1,y0,›1›);
img1.Canvas.TextOut(x0,y0–mush2,›1›);
img1.Canvas.TextOut(x0+(mush1 div
4),y0,›0›);
img1.Canvas.TextOut(x0–(mush1 div
2),0,›y›);
img1.Canvas.TextOut(ClientHeight-
mush1,y0+(mush2 div 4),›x›);
```

Данный код позволяет построить координатные оси, представленные на рисунках 2 и 3. Для построения координатной сетки используем цикл For:

```
for i:= 0 to ClientWidth do
begin
img1.Canvas.MoveTo(0,y0+i * mush2);
img1.Canvas.LineTo(ClientWidth,y0+i*
mush2);
end;

for i:= 0 to ClientWidth do
begin
img1.Canvas.MoveTo(0,y0–i * mush2);
img1.Canvas.LineTo(ClientWidth,y0–i *
mush2);
end;

for i:= 0 to ClientHeight do
begin
img1.Canvas.MoveTo(x0+i * mush2,0);
img1.Canvas.LineTo(x0+i *
mush2,ClientHeight);
end;

for i:= 0 to ClientHeight do
begin
img1.Canvas.MoveTo(x0–i * mush2,0);
img1.Canvas.LineTo(x0–i *
mush2,ClientHeight);
end;
```

В данном коде параметры ClientWidth и ClientHeight являются максимальными размерами для объекта Timage, то есть это его длина и высота в пикселях.

Для удобства работы также воспользуемся кнопками «Очистить» и «Выход», которые позволяют сбросить все введенные значения в поля Tedit и закрыть запущенную программу.

Для корректной работы разработанного приложения на поля Tedit введем ограничения по вводу значений, которые будут запрещать ввод буквенных выражений и десятичных дробей:

```
procedure TForm1.edt1KeyPress(Sender:
TObject; var Key: Char);
begin
case Key of
‹1›..›9›,#8:;
‹0›: if edt1.SelStart=0 then Key :=#0;
‹-›: if (edt1.SelStart=0)and(Pos(‹-›,edt1.
Text)=0) then key:=#45 else key:=#0;
#1..#7,#9..#44,#46,#47,#58..#255:
key:=#0;
end;
end;
```

Данная процедура запрещает вводить какие-либо иные символы кроме 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и знака «-».

Полный код разработанной программы представлен в приложении 1. В приложениях 2 и 3 представлен результат работы разработанной программы.

Заключение

Работая над научно-исследовательской работой, мне пришлось изучить учебно-методическую и научную литературу, а также проанализировать материал, изученный мной в школе.

Особенно мне понравилось разрабатывать приложение для построения графиков элементарных функций при помощи Borland Delphi 7, так как при его создании возникало много ошибок и проблем, которые приходилось решать, прибегая к помощи преподавателей и одноклассников.

Целью данной работы было создание приложения для построения графиков элементарных функций, изучаемых в средней школе.

Хоть материал работы и выходит за рамки школьной программы, но он может изучаться в средней и старшей школе.

Я считаю, что цели работы достигнуты, а задачи решены. Результатом решения поставленных задач является приложение программный код которого приведен в приложении 1.

Разработанное приложение позволяет строить графики линейных и квадратичных функций по их коэффициентам. Продолжая работать над данным приложением, есть возможность расширить его функционал и уйти от построения графиков по коэффициентам и строить графики любых элементарных функций по их записи. Достижение этой цели требует более глубокого изучения функциональных возможностей программы Borland Delphi 7 и более глубокого освоения языка программирования Pascal.

Список литературы

1. Архангельский А.Я. Программирование в Delphi 7. – М.: ООО «Бином-Пресс», 2003 г. – 1152 с.: ил. ISBN 5–9518–0042–0