МБОУ «Ковылкинская средняя общеобразовательная школа №4»

Название работы: Графики функций (лабораторный практикум)

Выполнил: Королёв Михаил Александрович, ученик 11 класса

Руководитель: Честнова Светлана Николаевна, учитель информатики и математики

Ковылкино
2014

Оглавление

Оглавление 2
Введение 3
Основная часть 4
Заключение 11
Список литературы 12

Введение

Что такое Lazarus? Среда для объектно-ориентированного программирования на языке Free Pascal. Визуально очень напоминает Delphi, но в отличии от нее Lazarus – свободная программа, используя ее вы не нарушаете ни чьих прав. Существует версия для Windows-систем, а также версии для MacOS и Linux. Как у любого программного средства имеет свои недостатки, в частности достаточно большой размер получаемых файлов, нестабильная работа на некоторых системах. Но в целом, для изучения ООП Lazarus подходит замечательно. Изучение объектно-ориентированного программирования мы в информационном профиле проходим в третий четверти одиннадцатого класса. Изучая данную тему, меня заинтересовал вопрос: «что создать полезного, как для меня, так и для других?». Это и послужило выбором темы моего проекта.

Тема работы: «Графики функций» (лабораторный практикум).

Гипотеза: Построение графиков функций одна из интереснейших тем в школьной математике.

Один из крупнейших математиков нашего времени Израиль Моисеевич Гельфанд писал: «Процесс построения графиков является способом превращения формул и описаний в геометрические образы. Это – построение графиков – является средством увидеть формулы и функции и проследить, каким образом эти функции меняются…»

Цель работы: создать программу в помощь учителю, при объяснении, и учащимся, при закреплении, темы «Графики функций».

Для достижения данной цели в работе решаются следующие задачи:

1. изучить теоретические сведения;
2. описать этапы работы;
3. разработать программу в визуальной среде программирования «Lazarus»;
4. произвести отладку программы;

5. сделать общий вывод о применении программы.

Актуальность работы состоит в том, что лабораторный практикум «Графики функций» можно применять в учебном процессе в средней школе. Он, безусловно, вызовет интерес учителей, будет способствовать закреплению умений и навыков учащихся, повысит познавательный интерес к изучению математики.

В работе использованы следующие методы:

1. анализ научных статей и публикаций по изучаемой теме;
2. разработка программы в визуальной среде программирования «Lazarus»;
3. апробация полученных результатов.

Практической значимостью данной работы является возможность применения данного материала на уроках математики.

Основная часть

Программирование – это научная дисциплина. Если бы процессы программирования разных задач не имели между собой ничего общего, то программирование, как таковое, не было бы научной дисциплиной. Но дело обстоит не так. Существуют общие методы, которые позволяют, постепенно расчленяя задачи на подзадачи, сводить их решение, в конечном счете, к некоторым элементарным, подобно тому, как разбирая совершенно непохожие сложные механизмы, мы обнаруживаем, что они состоят из одинаковых деталей и узлов, только по-разному соединенных. Из этого, конечно, не следует, что процесс программирования не содержит в себе элементов творчества. Составление программы, такой же творческий процесс, что и разработка сложного механизма из заданных наборов деталей.

Процесс решения задачи на компьютере состоит из ряда этапов, включающих как подготовку задачи к решению, так и собственно решение.

Эти этапы включают:

1. постановку задачи;

2. построение модели изучаемого процесса или явления;

3. выбор метода решения;

4. разработка алгоритма решения задачи;

5. составление программы (кодирование);

6. отладка и тестирование программы;

7. собственно вычисления;

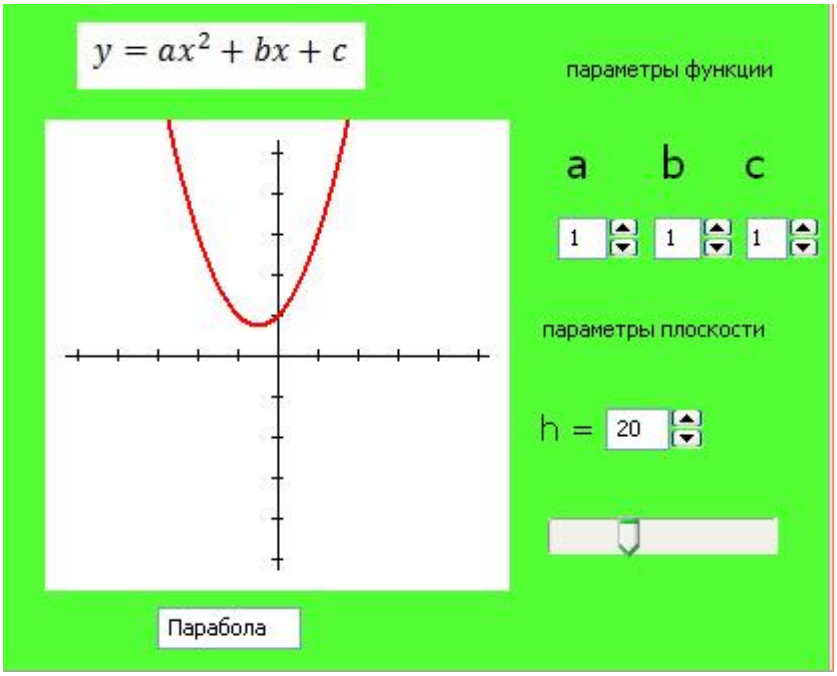
8. анализ полученных результатов.

Если первые три этапа были описаны во введении работы, то остановимся на последующих. Под составлением программы подразумевается непосредственная запись полученных ранее алгоритмов на выбранном языке программирования. Современные системы программирования позволяют значительно облегчить этот процесс, хотя этот этап по-прежнему остается одним из самых трудоемких. Разумеется, этот этап предполагает хорошее знание того языка программирования, на котором ведутся работы.

Программирование данной работы выполнено в визуальной среде программирования «Lazarus».

В лабораторном практикуме построены шесть графиков функций.

В качестве примера рассмотрим алгоритм построения графика квадратичной функции



Алгоритм выполнения:

1. создание и настройка визуальных компонентов
2. создание процедуры для рисования координатной плоскости
3. создание процедуры для рисования точек на плоскости
4. добавление возможности изменять параметры и масштабировать график

Для построения изображения использовались следующие компоненты:

1. TImage (на нем строится график)
2. текстовые поля TEdit для ввода параметров кубической функции и масштаба координатной плоскости
3. надписи на форме Label
4. визуальный компонент для масштабирования графика TrackBar
5. визуальные компоненты для изменения параметров текстовых полей UpDown

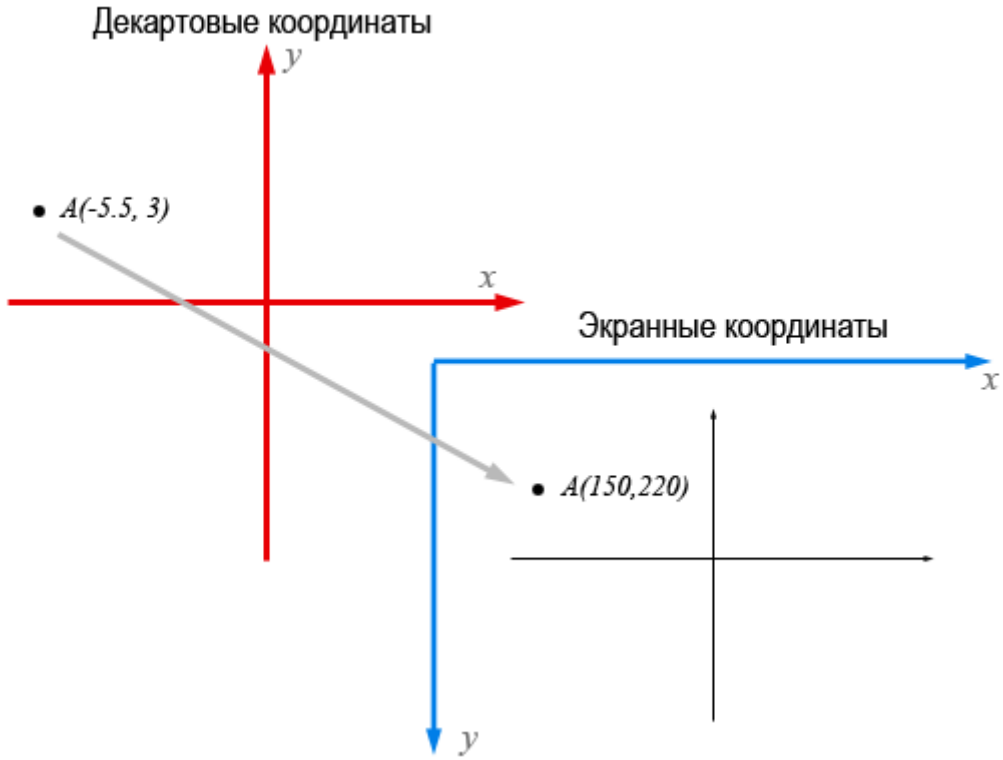
Построение координатных осей

Так как построение осей происходит в программе многократно, например при изменении параметров функции или при масштабировании, то имеет смысл вынести это действие в отдельную процедуру.

```
 procedure decart;
begin
  //задаем цвет карандаша: черный
  Form1.image1.Canvas.pen.color:= clBlack;
  // определяем ширину и высоту поля изображения
  w := Form1.image1.Width;
  h := Form1.image1.Height;
  // Находим координату нулевой точки
  x0 := w div 2;
  y0 := h div 2;
  // извлекаем из текстового поля m его значение,
  // это масштаб - длина единичного отрезка на плоскости (в px)
```

```
dd:= StrToInt(Form1.m.Text);
// так как дальнейшие операции совершаются со свойством Canvas объекта Image1,
// эта команда упрощает обращение к свойству
with Form1.image1.Canvas do
  begin
  Clear;
  pen.width:= 1;
   // строим координатную плоскость
   moveto(x0, 10);
   lineto(x0, h - 10);
   moveto(10, y0);
   lineto(w - 10, y0);
   n := x0 div dd;
   // строим деления плоскости
   for i := 0 to n do
     begin
     moveto(x0 + i * dd, y0 - 3);
     lineto(x0 + i * dd, y0 + 3);
     moveto(x0 - i * dd, y0 - 3);
     lineto(x0 - i * dd, y0 + 3);
     moveto(x0 - 3, y0 + i * dd);
     lineto(x0 + 3, y0 + i * dd);
     moveto(x0 - 3, y0 - i * dd);
    lineto(x0 + 3, y0 - i * dd);
    end;
  end;
end;
```

Для построение точек на экране компьютера нам многократно нужно совершать перевод декартовых координат в экранные. Напомню, что на экране координатные оси располагаются иначе, ось у «смотрит» вниз, к тому же экранная координата, это всегда целое положительное число, а декартовая координата может быть произвольным числом.



Перевод декартовых координат в «экранные». Значение координат точки взяты произвольные.

Преобразование выполняется следующим образом:

хэ := x0 + round(хд*dd);

уэ := y0 - round(уд*dd);

x0, y0 - экранная координата нулевой точки

dd - количество точек на экране, которое соответствует единичному отрезку (длина единичного отрезка).

Процедура рисования точек на экране

```
procedure graph;
begin
```

```
//задаем цвет карандаша: красный
Form1.image1.Canvas.pen.color:= clRed;
// минимальное и максимальное значение x на экране в текущем масштабе
x := -(x0/dd);
x_max := x0/dd;
// декартова координата начальной точки графика
y := a*x*x + b*x + c;
// экранные координаты первой точки
x1 := x0 + round(x*dd);
y1 := y0 - round(y*dd);
// параметры кубической функции
a:= strtofloat(Form1.Edit1.Text);
b:= strtofloat(Form1.Edit2.Text);
c:= strtofloat(Form1.Edit3.Text);
 // Построение точки от точки до точки непрерывного графика строится отрезок с помощью встроенной процедуры moveto()
Form1.image1.Canvas.pen.width:= 2;
Form1.image1.Canvas.moveto(x1, y1);
while x<x_max  do
begin
 // получение очередной декартовой точки
 y := a*x*x + b*x + c;
 // переводим декартову в экранную координату
 x1 := x0 + round(x*dd);
 y1 := y0 - round(y*dd);
 // строим отрезок до очередной экранной точки
 Form1.image1.Canvas.lineto(x1, y1);
 // шаг дакартовой точки с которым мы делаем построение графика
 x := x+0.1;
```

```
   end;
end;
```

TrackBar

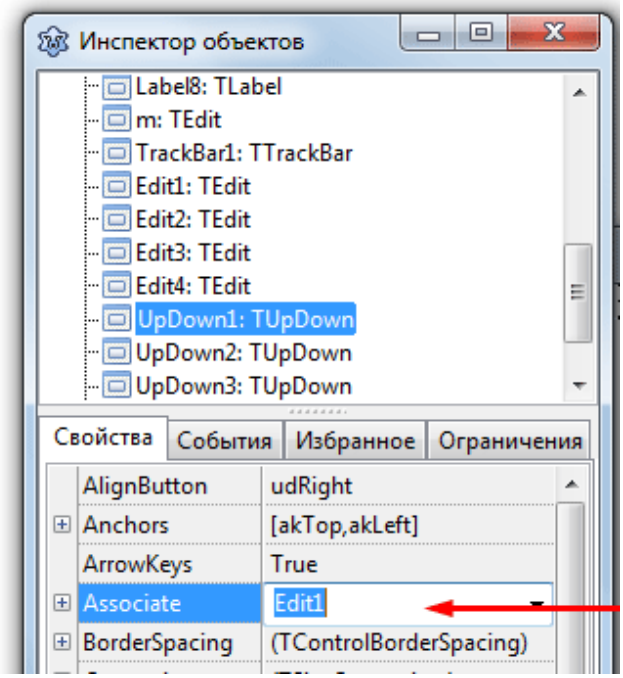
Масштабирование графика

Масштабирование графика функции сводится к изменению значения текстового поля m и перерисовки экрана. Для этого используется визуальный компонент TrackBar. Утановите компонент на форме, задайте его свойства min, max, position (минимальное, максимальное значение, которое может принять компонент, а также текущее значение масштаба). Создайте событийную процедуру, что должно происходить при изменении позиции ползунка:

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  // помещаем новое значение ползунка в текстовое поле
  m.Text:= IntToStr(TrackBar1.Position);
  // перерисовывем экран и график (вызываем соответствующие процедуры
  decart;
  graph;
end;
```

Изменение параметров функции

Значения параметров a, b. c определяют вид графика функции. Для изменения текстовых полей, в которых хранятся значения используем визуальные компоненты UpDown их можно привязать к текстовому полю (свойство Associate). При этом само текстовое поле можно закрыть для редактирования (свойство ReadOnly - true).



При нажатии на кнопки вверх-вниз значение текстового поля изменяется на единицу и срабатывает событийная процедура:

```
procedure TForm1.UpDown1Click(Sender: TObject; Button: TUDBtnType);
begin
 // перерисовка плоскости и графика
  decart;
  graph;
end;
```

Подобно можно построить любой график функции.

Вот и все. Проект необходимо отладить, собрать (получить запускаемый файл). С файлами проекта и с компилированным файлом вы можете ознакомиться во вложении к этому материалу.

Заключение

Подводя итог работы, хотелось бы сделать вывод: цель, поставленная в начале выполнения проекта, была достигнута. Мною разработанный лабораторный практикум «Графики функций» можно применять в учебном процессе в средней школе. Гипотеза поставленная мною, верна. В этом я убедился, когда предложил учителям нашей школы применять мою программу при изучении графиков функции. Желающим написать подобную программу, нужно придерживаться алгоритма, описанного мною в практической части данной работы.

Список литературы

1. Копченова Н.В., Марон И.А. "Вычислительная математика в примерах и за-дачах", М.: "Наука", 1972.
2. Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К. "Объектно-ориентированное программирование", М.: Изд-во МГТУ им. Н.Э. Баумана, 2003.
3. Сайт FreePascal.ru – http://www.freepascal.ru/
4. Форум сообщества ALT Linux – http://forum.altlinux.org/
5. Форум программистов и сисадминов – http://www.cyberforum.ru/
6. http://freepascal.org/
7. http://lazarus.freepascal.org/