Государственное образовательное учреждение дополнительного профессионального образования (повышения квалификации) работников образования Московской области

(ГОУ Педагогическая академия)

Кафедра информационно-коммуникационных технологий

Итоговый практико-значимый проект

Методика преподавания темы «Основы объектно-ориентированного программирования»

по курсу вариативного учебного модуля

«Методика преподавания углубленного курса алгоритмики и программирования с учетом требований новых образовательных стандартов и внешней оценки качества подготовки выпускников по информатике (ЕГЭ)»

Слушатель:
Кривко-Красько Сергей Васильевич,
учитель информатики
МОУ «Лицей № 26» г. Подольска

Преподаватель:
Кащей Владимир Васильевич,
доцент, кандидат педагогических наук

Академия
2013

Оглавление

Введение

Преподавание основ алгоритмизации и программирования в школьном курсе информатики и ИКТ не регламентировано достаточно строго нормативными документами, определяющими содержание предмета, не только на базовом, но и на профильном уровне. В «Стандарте среднего (полного) общего образования по информатике и ИКТ» (1) отсутствует какое-либо упоминание об основах объектно-ориентированного программирования, поэтому степень знакомства учащихся с ООП зависит прежде всего от учителя. Учебники различных авторов предлагают различный подход к изучению этой темы. Так, в учебнике Н.Д.Угриновича (2) после очень краткого введения в ООП и объяснения основных понятий следует ряд примеров-проектов, созданных в системах объектно-ориентированного программирования Delphi, Visual Basic и др. В учебнике И.Г.Семакина и др. (3) также присутствует краткое объяснение базовых понятий ООП, далее следует знакомство с системой программирования Delphi и разобран ряд проектов по моделированию различных процессов. В учебнике К.Ю.Полякова и Е.А. Еремина (4) – серьезный теоретический материал по ООП, знакомство с системой программирования Lazarus (свободное программное обеспечение), но все это не подкреплено какой-либо практической деятельностью.

В таких условиях каждому учителю, желающему познакомить учащихся с основами ООП, приходится выбирать свой путь.

На мой взгляд, обучение основам программирования в профильном курсе информатики (у меня это классы физико-математического и информационно-технологического профиля) необходимо завершать знакомством с системами визуального программирования. А т.к. обучение программированию проходит на языке Паскаль, то такой системой программирования является, естественно, Delphi.

Следует обратить внимание, что ни один из вышеназванных учебников, а также ни одно из известных пособий, рассчитанных на обучение програм-

мированию старшеклассников, таких как книги С.Окулова (5), Н.Культина (6), (7) А.Желонкина (8), не содержат задач на программирование с созданием собственных классов (что можно было бы сделать, не используя системы визуального программирования, а опираясь только на Pascal Object). Поэтому содержание таких базовых понятий ООП как инкапсуляция, наследование, полиморфизм, остается «за кадром», т.е. не может быть реально усвоено. И это естественно, т.к. разработка таких программ вряд ли возможна на школьном уровне. Остается использование компонент готовых классов и фактически преподавание темы сводится к знакомству с возможностями создания программ с современным графическим интерфейсом.

В то же время использование систем визуального программирования без хотя бы минимального усвоения таких понятий как «объект», «свойство», «метод» невозможно. Поэтому практической деятельности неизбежно предшествует краткое теоретическое введение. Следует также отметить, что такое теоретическое введение неизбежно опирается на знание таких понятий как «подпрограмма», «комбинированный тип данных (запись)».

Практические задания по ООП в среде разработки Delphi, на мой взгляд, должны быть таковы, чтобы не возникало проблем с разработкой алгоритма, а усилия были сосредоточены в основном на разработке интерфейса.

Здесь можно выделить несколько этапов работы, прохождение которых зависит от возможности выделить на эту тему то или иное количество часов:

начальные проекты; несложные задачи с использованием минимального круга компонентов;

практические задания с использованием того же круга компонентов, но требующие большего объема работы;

задания на использование ряда дополнительных компонент;

задания на программирование графики;

задания на использования списков;

создание программ с мультимедийными компонентами.

Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) является в настоящее время наиболее популярной технологией программирования. Объектно-ориентированными системами программирования являются Visual Basic, Delphi (развитие языка программирования Pascal). В среде Microsoft Office существует встроенный язык программирования Visual Basic for Application (*перевод - «для приложений»*), посредством которого можно программировать макросы - действия по обработке данных в Microsoft Word, Microsoft Excel и в других программах Microsoft Office.

Объектно-ориентированное программирование (ООП) — это технология, основанная на представлении программ в виде совокупности объектов, каждый из которых является реализацией собственного класса, которые в свою очередь образуют иерархию на принципах наследования.

При использовании технологии ООП решение представляется в виде результата взаимодействия отдельных элементов некоторой системы, имитирующей процессы, происходящие в предметной области поставленной задачи. Каждый элемент системы, получая сообщение, выполняет заранее определенную последовательность действий (например, обрабатывает полученные данные, изменяет свое состояние, пересылает полученные данные другому элементу системы). Передавая сообщения от одного элемента системы к другому, система выполняет поставленную перед ней задачу.

Элементы системы, параметры и поведение которой определяются условием задачи, обладающие самостоятельным поведением (т. е. «умеющие» выполнять некоторые действия, зависящие от полученных сообщений и состояния элемента), получили название объектов. (5)

Основные понятия ООП

ООП характеризуется четырьмя основополагающими идеями - абстрагирование, инкапсуляция, наследование, полиморфизм.

Абстрагирование — это один из главных способов решения сложных задач. В результате объектной декомпозиции выделены объекты. Абстракция предназначена для выделения существенных характеристик каждого объекта, отличающих его от всех других видов объектов и, таким образом, четко определяются его концептуальные границы с точки зрения наблюдателя.

Для представления абстракций объектов используется специальный определяемый программистом тип данных — **класс.**

Класс — это структурный тип данных, который включает описание полей данных, а также процедур и функций, работающих с этими полями данных. Класс объединяет в себе **поля** (описывающие его данные - **свойства)**, так и средства обработки этих данных – **методы**. Другими словами, свойства – это переменные какого-либо типа, а методы – подпрограммы их обработки.

Процесс объединения данных с действиями над этими данными в единый пакет при наличии специальных правил доступа к элементам пакета получил название **инкапсуляция.**

Итак, сочетание данных с допустимыми действиями над этими данными приводит к «рождению» нового «кирпичика» программирования — класса.
Класс представляет собой структуру, динамически размещаемую в памяти.
Каждый отдельный **объект** является экземпляром своего класса, и он наследует весь набор полей и методов этого класса. Например, в приложении Microsoft Word существует класс объектов «символ», который обладает свойствами: шрифт, размер, цвет, начертание и т.п. и значит, у каждого конкретного символа, являющегося экземпляром этого класса, есть эти свойства и они имеют конкретные значения. Аналогично – классы «документ», «абзац», «фрагмент».

В программе Delphi существуют классы Form, Button, Edit и другие. Создавая объект – экземпляр какого-либо класса, мы получаем совокупность переменных, определяющих свойства этого объекта, значения которых можем в программе задавать, изменять, использовать.

Каждый объект, созданный в программе, получает свое уникальное имя. Их свойства обозначаются следующим образом: ИмяОбъекта.Свойство.

Аналогично описываются методы объекта:

ИмяОбъекта.Метод(Список формальных параметров).

Наследование — это процесс, посредством которого один объект может наследовать свойства другого объекта и добавлять к ним черты, характерные только для него. В итоге создаётся иерархия объектных типов, где поля данных и методов "предков" автоматически являются и полями данных и методов "потомков".

Смысл и универсальность наследования заключается в том, что не надо каждый раз заново ("с нуля") описывать новый объект, а можно указать "родителя" (базовый класс) и описать отличительные особенности нового класса. В результате новый объект будет обладать всеми свойствами родительского класса плюс своими собственными отличительными особенностями.

Полиморфизм — это свойство, которое позволяет одно и тоже имя использовать для решения нескольких технически разных задач. Полиморфизм подразумевает такое определение методов в иерархии типов, при котором метод с одним именем может применяться к различным родственным объектам. В общем смысле концепцией полиморфизма является идея "один интерфейс — множество методов". Преимуществом полиморфизма является то, что он помогает снижать сложность программ, разрешая использование одного интерфейса для единого класса действий. Выбор конкретного действия, в зависимости от ситуации, возлагается на компилятор.

Событие – это действие, распознаваемое объектом. Событие может создаваться пользователем программы, например, щелчок мышью или нажа-

тие клавиши. Системные события исходят непосредственно от операционной системы (например, открытие или закрытие формы. Оно может быть результатом выполнения программы, например, попадание объекта в некоторую область на экране. В принципе, событием может быть изменение любой величины. Событие заставляет работать определенный объект. В качестве реакции на событие вызывается некоторая подпрограмма (процедура), называемая - **обработчик события**, которая может изменять свойства этого или какого-либо другого объекта, вызывать его методы и т.п.

Имя такой процедуры включает имя объекта и имя события. Например, Button1Click.

Системы визуального программирования

Системы визуального программирования – это разновидность систем объектно-ориентированного программирования (существуют системы ООП, не являющиеся визуальными, например, ObjectPascal, C++).

Их особенностью является возможность визуального использования объектов графического интерфейса, т.е. возможность создавать объекты и задавать значения их свойств не только путем написания соответствующих команд в программе, а также путем размещения их на форме.

Основными объектами при визуальном программировании являются формы (Form), на которых размещаются управляющие элементы (объекты):

Командные кнопки (Button), текстовые поля (Edit), графические окна (PaintBox), горизонтальные и вертикальные полосы прокрутки (ScrollBar), переключатели (RadioButton) и другие.

Программирование в основном состоит в размещении объектов на формах, задании значений их свойств и написании событийных процедур (т.е. подпрограмм, выполняющихся при возникновении события для данного объекта, например, щелчка мышки, протяжке полосы прокрутки).

Разрабатываемая на языке визуального программирования программа

(приложение) называется проектом. Проект включает в себя не только форму со всеми своими элементами, но и программные модули событийных процедур. В результате разработки проекта может быть создан файл-программу с расширением .exe, который потом можно запускать на выполнение без системы программирования.

Отработка заданий по визуальному программированию возможна как в среде программирования Delphi, так и в PascalABC. Использование PascalABC значительно проще (легкий интерфейс, минимальное количество компонент и свойств объекта), но имеет ряд недостатков:

нельзя создать итоговый exe-файл;

не все компоненты из имеющихся можно использовать, программа явно не доработана.

Поэтому далее задания рассматриваются применительно к Delphi, хотя некоторые из них выполнялись и в PascalABC.

При разработке заданий использовались пособия Н.Культина (6), (7), А.Желонкина (8), а также сайты «Уроки Delphi начинающим с нуля» (9), «100 компонентов Delphi» (10).

Основные компоненты Delphi

Ниже приведены основные свойства наиболее используемых компонент.

Класс (компонента)	Свойство	тип	назначение
любой	Name	string	Имя объекта (имя переменной)
	Left	integer	Горизонтальная координата верхнего левого угла
	Top	integer	Вертикальная координата верхнего левого угла
	Width	integer	Ширина
	Height	integer	Высота
	Color	integer	Цвет фона
	Caption	string	Текст заголовка или надписи
	visible	boolean	Видимость элемента
Form			Форма (стандартное окно приложения)
Button			Кнопка
Edit			Поле ввода
	Text	string	Строка в поле ввода
	Enable	boolean	Возможность ввода строки с клавиатуры
Label			Метка (объект для отображения текста)
Scrollbar			Полоса прокрутки
	Min	integer	Минимальное значение бегунка
	max	integer	Максимальное значение бегунка
	position	integer	Текущее значение бегунка
	Kind		Ориентация (верт. или горизонт.)
RadioButton			Зависимая кнопка (входящая в группу)
	Checked	boolean	Состояние выбора (true-выбрана, false-нет)
CheckBox			Независимая кнопка (флажок)
	Checked	boolean	Состояние выбора (true-выбрана, false-нет)
ListBox			Список
	Items		Элементы списка – массив строк
	Count		Количество элементов списка
	ItemIndex		Номер выбранного элемента
ComboBox			Выпадающий список
	Text		Текст в поле ввода
	Items		Элементы списка – массив строк
	Count		Количество элементов списка
	ItemIndex		Номер выбранного элемента

Графика на Delphi

Некоторые компоненты (Form, Image, PaintBox) обладают свойством Canvas («холст»), представляющим собой поверхность для вывода графики.

Для того чтобы на поверхности объекта появился графический элемент, необходимо к свойству Canvas этого объекта применить соответствующий метод: **Имя_объекта.canvas.метод (параметры).**

Координаты выводимых линий, фигур определяются относительно самого объекта. Точка с координатами (0,0) находится в верхнем левом углу объекта. Размер области определяется свойствами Width, Height.

Основные методы Canvas:

LineTo(x,y) – вычерчивает линию из текущей точки в указанную;

MoveTo(x,y) – перемещает указатель текущей точки в указанную;

Rectangle(x1,y1,x2,y2) – прямоугольник с заданными координатами противоположных углов;

Ellipse(x1,y1,x2,y2) – эллипс, вписанный в прямоугольник с заданными координатами противоположных углов;

Arc(x1,y1,x2,y2,x3,y3,x4,y4) – дуга эллипса, определенного параметрами x1,y1,x2,y2, параметры x3,y3,x4,y4 задают координаты точек, лучи от которых до центра эллипса отсекают дугу, дуга вычерчивается против часовой стрелки от луча к (x3,y3) до луча к (x4,y4).

TextOut(x,y,s) – выводит строчку s от точки с заданными координатами.

FloodFill (x,y, цвет, стиль) – заливка от указанной точки, если стиль – fsborder, то заливка до границы указанного цвета, если стиль fsSurface - то перекраска области указанного цвета.

Дополнительно (PascalABC):

Line(x1,y1,x2,y2) –отрезок с началом в точке (x1,y1) и концом в точке (x2,y2).

Circle(x,y,r) окружность с центром в точке (x,y) и радиусом r

Свойства Canvas

Свойство **Pen** определяет цвет, тип, толщину выводимых линий.

У него есть свои свойства:

Color - цвет; **Width** - толщина;

Style – тип линии (psSolid – сплошная линия, используется по умолчанию; psDash – пунктир из длинных штрихов; psDot – пунктир из коротких штрихов; psDashDot – пунктир из коротких и длинных штрихов; psClear – не отображается);

Свойство **Brush** определяет цвет и стиль закрашиваемой области.

Его свойства:

Color – цвет;

Style – стиль заполнения области (bsSolid – сплошная заливка, используется по умолчанию; bsClear – область не закрашивается; bsHorisontal, bs Vertical, bsFDiagonal, bsBDiagonal - ; bsCross, bsDiagCross – различные штриховки).

Задание цвета

Цвет может задаваться через константы типа TColor, например, clBlack, clWhite, clRed, clBlue, clGreen, clYellow, clgGray и т.п.

Также для задания цвета можно использовать функцию RGB(r,g,b), где параметры r, g, b задают степень интенсивности каждой из трех компонент цвета (значения от 0 до 255).

При использовании свойств и методов приходится писать многоступенчатые имена, например, Paintbox1.canvas.pen.color, Paintbox1.canvas.lineto и т.д.

Для упрощения записи можно использовать оператор With:

```
With paintbox1.canvas do
begin
pen.color:=…
Lineto(…);
….
end;
```

Практические задания по ООП в Delphi

Как сказано выше, разработано несколько типов заданий, поэтапно осваиваемых.

Начальные проекты на Delphi

Этот цикл заданий предназначен для освоения интерфейса программы Borland Delphi и требует знания лишь простейших компонент : Form, Button, Edit, Label, а также функций перевода числа в текст и обратно IntToStr, FloatToStr, StrToInt, StrToFloat.

1. «Заголовок»

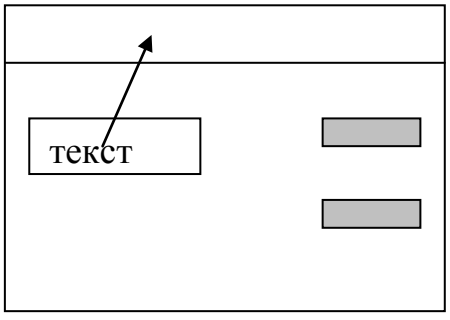
Расположить на форме кнопку Button и текстовое окно Edit. Написать программу, которая при нажатии кнопки переносит текст из текстового окна в заголовок формы (текстовое окно при этом очищается). Поэкспериментировать с различными свойствами текстового окна (цвет, шрифт). Добавить кнопку выхода из программы.

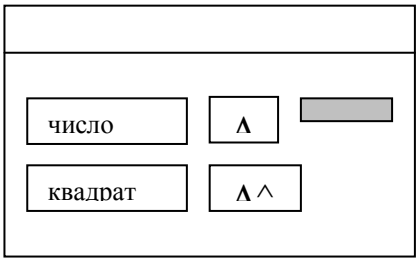
2. «Возведение в квадрат»

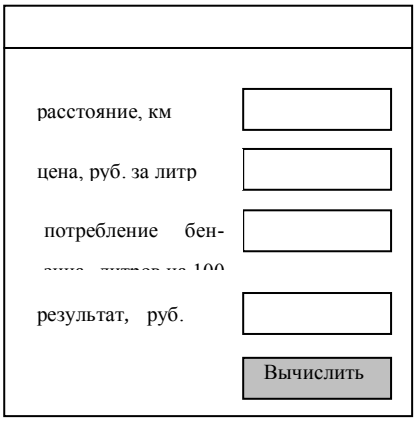
Расположить на форме два текстовых окна и кнопку. В одно текстовое окно вводится целое число (лучше тип longint). При нажатии кнопки в другом текстовом окне выводится квадрат этого числа. Использовать метки для комментирующих надписей.

3. «Поездка на автомобиле»

Написать программу, которая вычисляет стоимость поездки на автомобиле по заданным значениям расстояния, цены бензина за литр и потребления бензина в литрах на 100 км.







Вычислительные проекты на Delphi

Задания этого цикла требуют несколько большего объема работы и позволяют проявить индивидуальный подход к интерфейсу программы.

4. Калькулятор»

На форме должны быть размещены компоненты «Текстовые окна» (два для ввода исходных данных, одно – для вывода результата), кнопки для выполнения арифметических операций, кнопка для очистки текстовых окон и выхода.

Допускается использовать только одно окно для ввода числа.

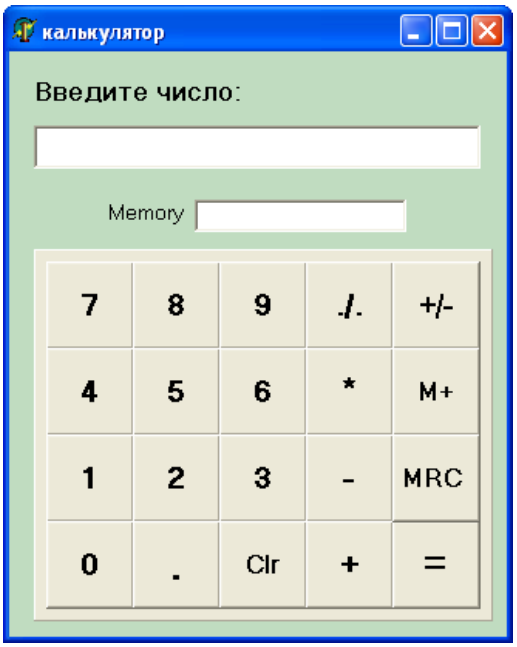
При делении на нуль в окне вывода должно быть соответствующее сообщение.

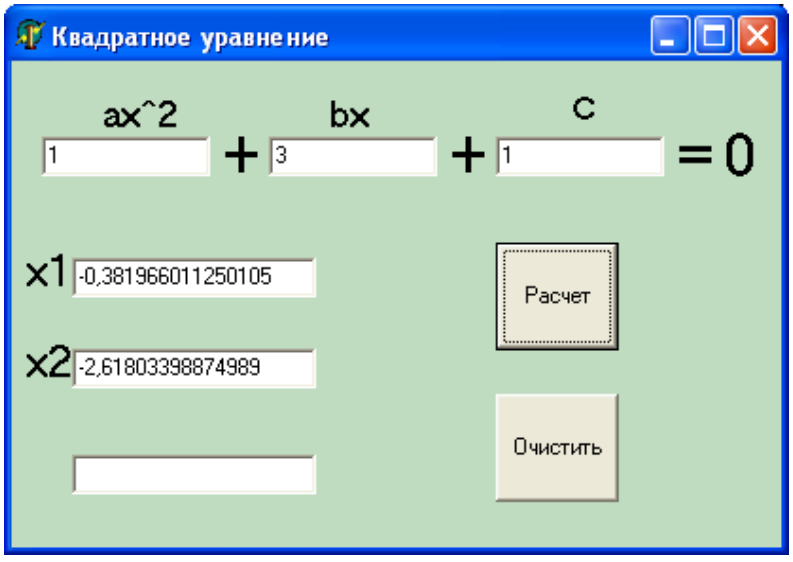
5. «Квадратное уравнение»

Программа должна решать квадратное уравнение $ax^2+bx+c=0$ при любых значениях a,b,c (в т.ч. при a=0).

На форме должны быть размещены компоненты «Текстовые окна» (три для ввода исходных данных, два или три – для вывода результата), кнопки для выполнения расчета, кнопка для очистки текстовых окон и выхода.

При выводе результатов предыдущие результаты должны автоматически быть очищены.





Задания на использование дополнительных компонент;

6. «Светофор»

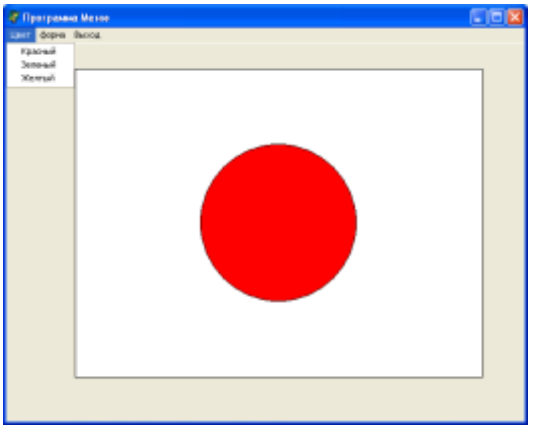
На форме должны быть расположены круг и переключатели RadioButton для выбора цвета – Красный, Желтый, Зеленый.

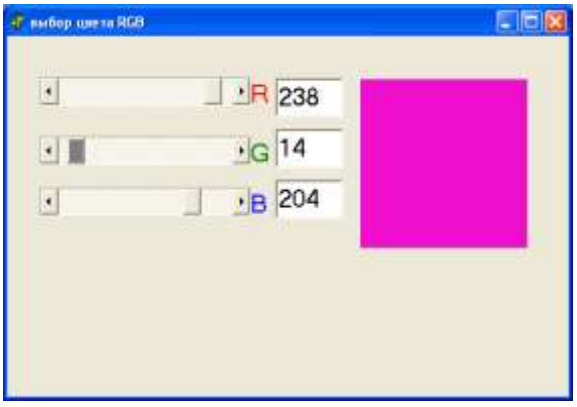
7. «Меню»

На форме располагается компонент – Shape (фигурв, форму которой можно задавать и невизуальный компонент MainMenu. Форма должна содержать меню из трех пунктов: Форма (для выбора формы фигуры – круг, прямоугольник, квадрат) Цвет (выбор цвета – несколько вариантов) и Выход.

8. «Выбор цвета – RGB».

На форме располагается прямоугольник и три полосы прокрутки для выбора значения трех составляющих цвета в модели RGB (значения от 0 до 255). Значения эти выводятся на экран. При перемещении любого из движков цвет прямоугольника изменяется на выбранный.





Задания на графику в Delphi

9. «Прямоугольный треугольник».

Ввод исходных данных (катеты) производится или в текстовые окна или же с помощью «полос прокрутки». При нажатии кнопки рисуется треугольник, вписанная и описанные окружности, при этом, если в поле рисования уже был рисунок, он должен быть стерт.

10. «График функции».

Параметры графика (количество отрезков на полуосях или масштабы осей) вводятся или в текстовые окна или же с помощью «полос прокрутки». При нажатии кнопки рисуется график функции (y=sin(x) или другой), оси, сетка, если в поле рисования уже был график, он должен быть стерт.

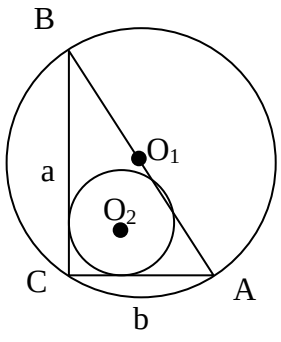
Разберем подробнее эти задания.

«Прямоугольный треугольник»

Описание метода построения и интерфейса программы

Ввод исходных данных (катеты) производится или в текстовые окна или же с помощью «полос прокрутки».

При нажатии кнопки рисуется треугольник, вписанная и описанные окружности, при этом, если в поле рисования уже был рисунок, он должен быть стерт.



Для разработки программы необходима некоторая математическая подготовка. Если у треугольника заданы катеты a,b, то радиус описанной окружности R вычисляется как c/2, где c – гипотенуза, вычисляемая по теореме Пифагора. Радиус вписанной окружности можно вычислить по формуле $r= S/p$, где S- площадь треугольника, p –полупериметр. Но необходимо еще и определить координаты вершин треугольника и центров этих

окружностей. Как один из возможных вариантов, можно фигуры расположить так, чтобы центр описанной окружности O_1 располагался в центре той компоненты, на которой будет выведено изображение. Если это будет PaintBox1, то тогда координаты этой точки будут: x0:= PaintBox1.width div 2, y0:= PaintBox1.height div 2.

Но тогда координаты точки A будут x0+b div 2; y0+a div 2,

Точки B – x0-b div 2; y0-a div 2, точки C - x0-b div 2; y0+a div 2.

Координаты точки O_2 будут x0-b div 2+r; y0+a div 2-r.

В формулах учтено, что направление координаты Y на экране сверху вниз.

При разработке программы предусмотрены две возможности ввода значений длин катетов – через текстовые окна и через полосы прокрутки, причем оба способа взаимосвязаны (при изменение движка на полосе прокрутки изменяется значение в текстовом окне и наоборот). Максимальные значения катетов ограничены размерами используемой компотенты PaintBox. Предусмотрены кнопки для вывода изображения треугольника, вписанной и описанной окружностей. Предусмотрена заливка каждой из фигур. При изменении значений катетов блокируются кнопки вывода окружностей и разблокируются при выводе треугольника. При выводе треугольника прежнее изображение стирается.

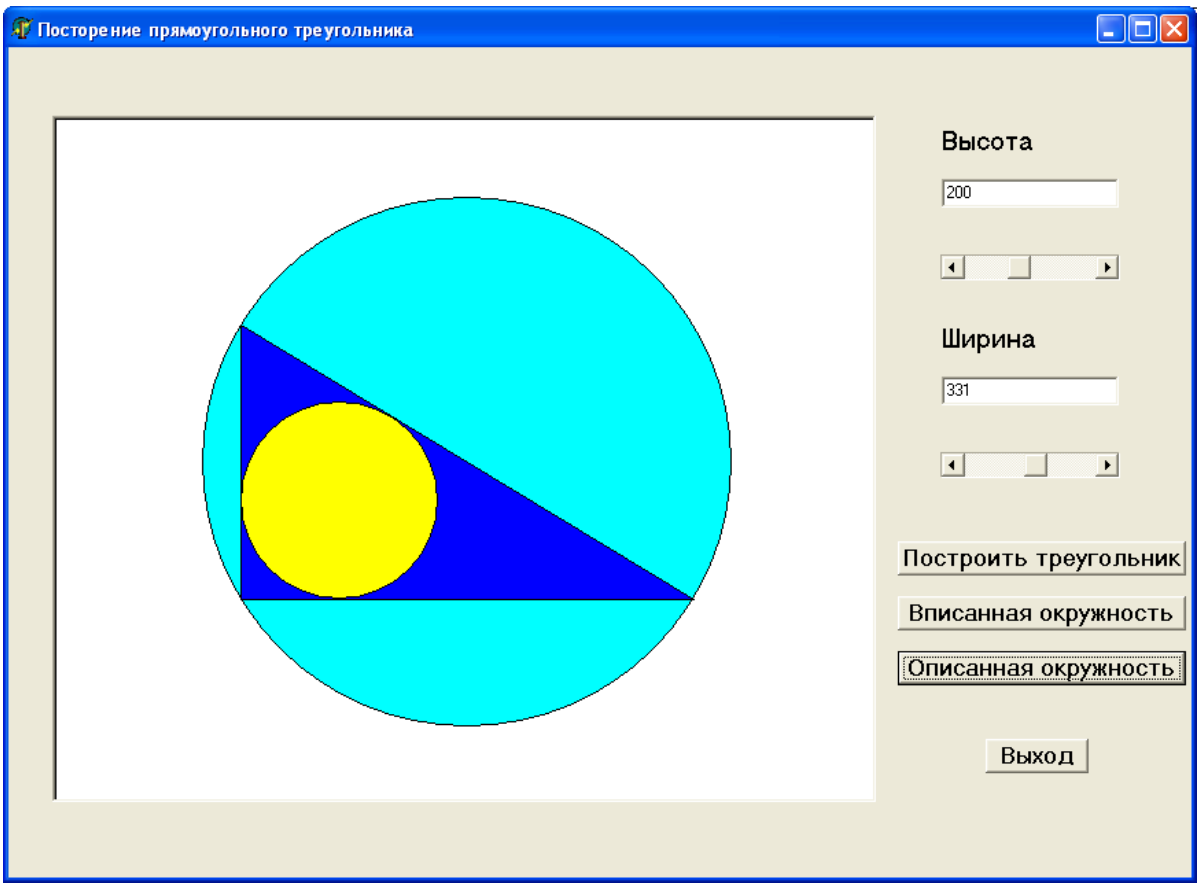
Листинг программы

```
unit treug;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Con-
trols, Forms,   Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    PaintBox1: TPaintBox;
    Button1, Button2, Button3, Button4: TButton;
    Edit1, Edit2: TEdit;
    ScrollBar1, ScrollBar2: TScrollBar;
    Label1, Label2: TLabel;
```

```
    procedure ScrollBar2Change(Sender: TObject);
    procedure ScrollBar1Change(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.ScrollBar2Change(Sender: TObject);
begin
button3.Visible:=false;button4.Visible:=false;
edit2.text:=inttostr(scrollbar2.Position);
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
button3.Visible:=false;
button4.Visible:=false;
edit1.text:=inttostr(scrollbar1.Position);
end;

procedure TForm1.Edit1Change(Sender: TObject);
var k:integer;
begin
button3.Visible:=false; button4.Visible:=false;
k:=strtoint(edit1.text);
if k>scrollbar1.max then
   begin k:=scrollbar1.max; edit1.text:=inttostr(k);
   end;
      scrollbar1.Position:=k;
end;
procedure TForm1.Edit2Change(Sender: TObject);
var k:integer;
begin
button3.Visible:=false; button4.Visible:=false;
k:=strtoint(edit2.text);
if k>scrollbar2.max then
   begin k:=scrollbar2.max;
         edit2.text:=inttostr(k);
   end;
      scrollbar2.Position:=k;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
form1.close;
end;
procedure TForm1.Button2Click(Sender: TObject);
var a,b,x0,y0:integer;
begin
button3.Visible:=true; button4.Visible:=true;
x0:=paintbox1.Width div 2;y0:=paintbox1.Height div 2;
a:=scrollbar1.Position div 2;b:=scrollbar2.Position div 2;
with paintbox1.canvas do
begin
pen.color:=clblack; brush.Color:=clwhite;
rectangle(0,0,2*x0,2*y0);
moveto(x0-b,y0+a);lineto(x0-b,y0-a);
lineto(x0+b,y0+a);lineto(x0-b,y0+a);
brush.Color:=clblue;
FloodFill(x0-1,y0+1,clblack ,fsborder );
end;
end;
procedure TForm1.Button3Click(Sender: TObject);
var a,b,x0,y0,r,xr,yr:integer; c:real;
begin
x0:=paintbox1.Width div 2; y0:=paintbox1.Height div 2;
a:=scrollbar1.Position; b:=scrollbar2.Position;
c:= sqrt(a*a+b*b);r:=trunc(a*b/(a+b+c));
xr:=x0-b div 2+r;yr:=y0+ a div 2 -r;
with paintbox1.canvas do
begin
pen.color:=clblack; brush.Color:=clyellow;
ellipse(xr-r,yr-r,xr+r,yr+r);
end;
end;
procedure TForm1.Button4Click(Sender: TObject);
var a,b,x0,y0,r:integer; c:real;
begin
x0:=paintbox1.Width div 2; y0:=paintbox1.Height div 2;
a:=scrollbar1.Position; b:=scrollbar2.Position;
c:= sqrt(a*a+b*b); r:=round(c/2);
with paintbox1.canvas do
begin
pen.color:=clblack; brush.Style:=bsclear;
ellipse(x0-r,y0-r,x0+r,y0+r);
brush.Style:=bssolid; brush.Color:=claqua;
FloodFill(x0+3,y0-3,clblack ,fsborder );
FloodFill(x0-b div 2-3,y0,clblack ,fsborder );
FloodFill(x0,y0+a div 2+3,clblack ,fsborder );
end;
end;
end.
```

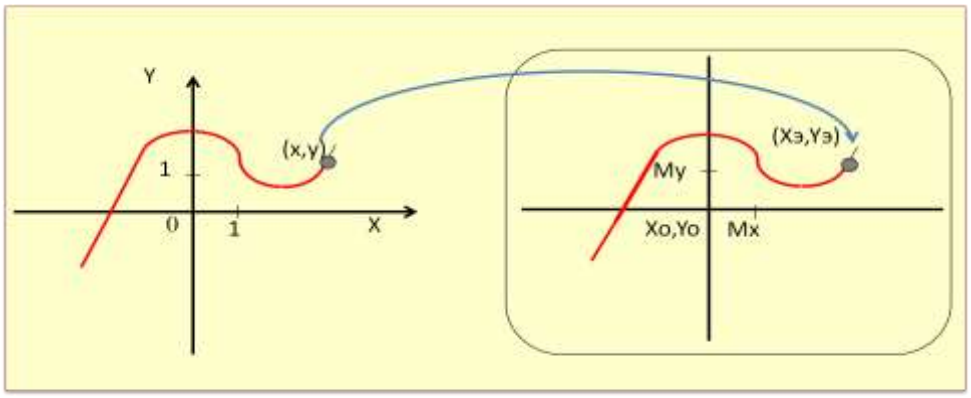
ScreenShot программы



«График функции»

Описание метода построения и интерфейса программы

Для построения графика функции предлагается следующий подход. Реальные (математические) координаты точек графика (x;y) пересчитываются в экранные (хэ;уэ) таким образом, чтобы точка начала координат (0;0) переходила в точку на экране с координатами (x0;y0), а единичные отрезки по осям X,Y переходил в отрезки Mx, My пикселей.



Реальные (математические) координаты

Экранные координаты

Формулы пересчета: $Xэ = X_0 + Mx * X$, $Yэ = Y_0 – My * Y$.

Тогда алгоритм построения графика функции можно представить в следующем виде (рис.1).

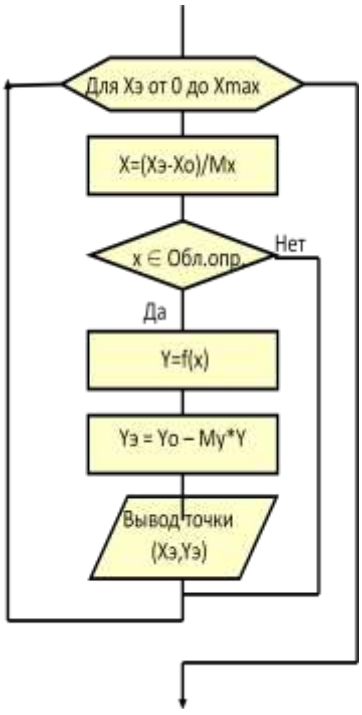


Рисунок 1

Общая схема построения графика функции:

- Ввод исходных данных (Xo, Yo, Mx, My).
- Построение координатных осей.
- Разметка осей. *(необязательно).*
- Построение графика по приведенному алгоритму.

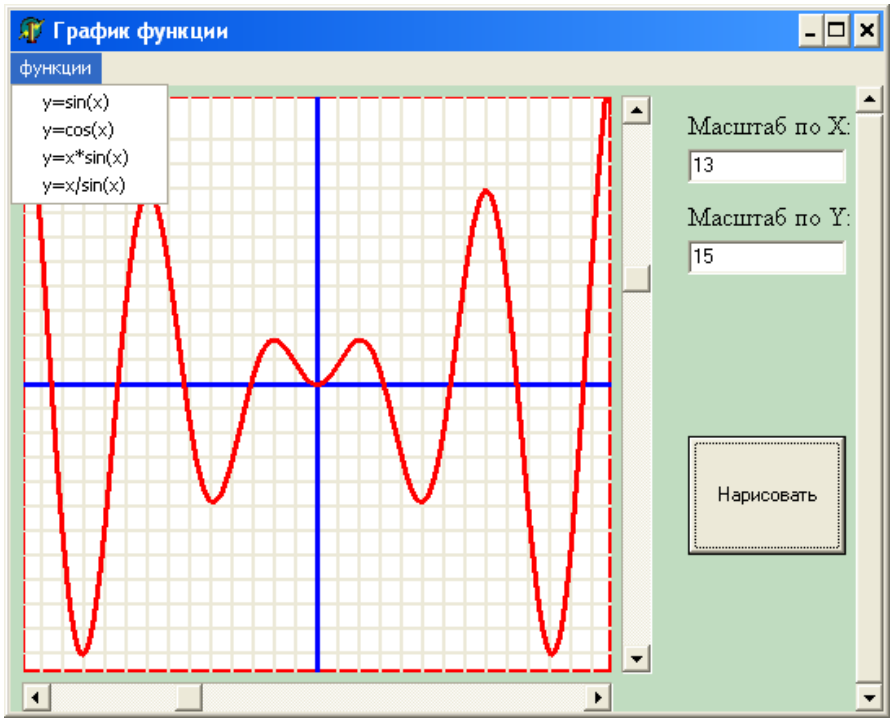
Листинг программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Con-
trols, Forms,
  Dialogs, StdCtrls, ExtCtrls, Menus;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    ScrollBar1, ScrollBar2: TScrollBar;
    pb: TPaintBox;
    Button1: TButton;
    Edit1, Edit2: TEdit;
    Label1, Label2: TLabel;
    MainMenu1: TMainMenu;
    a1: TMenuItem;
    ysinx1, ycosx1, yxcosx1, yxsinx1: TMenuItem;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ScrollBar2Change(Sender: TObject);
    procedure ScrollBar1Change(Sender: TObject);
    procedure ysinx1Click(Sender: TObject);
    procedure ycosx1Click(Sender: TObject);
    procedure yxcosx1Click(Sender: TObject);
    procedure yxsinx1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  nf:integer;
implementation
{$R *.dfm}
function f(x:real):real;
begin
if nf=1 then f:=sin(x);
if nf=2 then f:=cos(x);
if nf=3 then f:=x*sin(x);
if (nf=4) and(sin(x)<>0) then f:=x/sin(x);
end;

procedure TForm1.Button1Click(Sender: TObject);
var nx,ny,i,j,x0,y0,mx,my,xe,ye:integer;  x,y:real;
begin
with pb.Canvas do
begin
```

```
pen.Width:=2;
Rectangle(0,0,pb.width,pb.Height);
mx:=scrollbar2.Position; my:=scrollbar1.Position;
edit1.Text:=inttostr(mx); edit2.Text:=inttostr(my);
x0:=pb.Width div 2; y0:=pb.height div 2;
nx:=x0 div mx; ny:=y0 div my;
Pen.Color:=clbtnface; Pen.Style:=psdot;
MoveTo(x0-mx*nx,0);
i:=x0-mx*nx; j:=y0-my*ny;
while i<pb.Width do
  begin  LineTo(i,pb.Height); i:=i+mx; MoveTo(i,0); end;
while j<pb.Height do
  begin
 LineTo(pb.Width,j); j:=j+my; pb.Canvas.MoveTo(0,j); end;
Pen.Color:=clblue; Pen.Width:=3; Pen.Style:=pssolid;
MoveTo(0,y0) LineTo(pb.Width,y0);
MoveTo(x0,0); LineTo(x0,pb.Height);
Pen.Color:=clred;
xe:=0; x:=(xe-x0)/mx; y:=f(x); ye:=y0-round(my*y);
MoveTo(xe,ye);
for xe:=1 to pb.Width do
   begin
   x:=(xe-x0)/mx;    y:=f(x);  ye:=y0-round(my*y);
   LineTo(xe,ye);
   end;
end;
end;

procedure TForm1.FormCreate(Sender: TObject);
var mx,my:integer;
begin
mx:=scrollbar2.Position; my:=scrollbar1.Position;
edit1.Text:=inttostr(mx); edit2.Text:=inttostr(my);
end;

procedure TForm1.ScrollBar2Change(Sender: TObject);
var mx:integer;
begin
mx:=scrollbar2.Position; edit1.Text:=inttostr(mx);
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
var my:integer;
begin
my:=scrollbar1.Position; edit2.Text:=inttostr(my);
end;

procedure TForm1.ysinx1Click(Sender: TObject);
begin
nf:=1;
end;
```

```
procedure TForm1.ycosx1Click(Sender: TObject);
begin
nf:=2;
end;
procedure TForm1.yxcosx1Click(Sender: TObject);
begin
nf:=3;
end;
procedure TForm1.yxsinx1Click(Sender: TObject);
begin
nf:=4;
end;

end.
```

ScreenShot программы



Заключение

Использование объектно-ориентированного программирования в процессе обучения школьников программированию имеет особое значение. Ведь только применяя такие среды программирования как Delphi можно создать программы с современным интерфейсом, похожим на тот, с которым учащиеся встречаются, используя любую программу, хоть учебную, хоть игровую. Таким образом, можно повысить интерес к программированию и создать дополнительные стимулы к обучению. Сами же основные понятия ООП могут быть освоены прежде всего через их практическое применение. Методический материал по программированию в Delphi обширен и разнообразен, что позволяет учителю выстроить методику преподавания с учетом индивидуальных особенностей учащихся и может быть использован не только в учебной деятельности на уроке, но и во внеклассной работе.

Список литературы

1. *Программы для общеобразовательных учреждений: Информатика. 2-11 классы / Составитель М.Н.Бородин.* М. : БИНОМ. Лаборатория знаний, 2007. стр. 448.

2 **Угринович Н.Д..** *Информатика и ИКТ: Учебник для 10 класса.* М. : БИНОМ. Лаборатория знаний., 2007. стр. 371.

3. **Семакин И.Г., Хеннер Е.К., Шестакова Л.В.** *Информатика и ИКТ. Профильный уровень: учебник для 11 класса.* М. : БИНОМ. Лаборатория знаний, 2012. стр. 350.

4. **Поляков К.Ю., Еремин Е.А.** *Информатика. Углублённый уровень: учебник для 10 класса в 2 ч. Ч. 2.* М. : БИНОМ. Лаборатория знаний, 2013. стр. 304.

5. **Окулов, С.М., Бабушкина И.А.** *Практикум по объектно-ориентированному программированию.* М. : БИНОМ. Лаборатория знаний, 2004. стр. 366.

6. **Культин Н.Б.** *Delphi в задачах и примерах.* СПб. : БХВ-Петербург, 2003. стр. 288.

7. **Культин Н.Б.** *Основы программирования в Delphi 8 для Microsoft.NET.* СПб. : БХВ-Петербург, 2004. стр. 400.

8. **Желонкин А.В.** *Основы программирования в интегрированной среде DELPHI. Практикум.* М. : БИНОМ. Лаборатория знаний, 2004. стр. 236.

9. http://www.delphi-manual.ru/index.php. [В Интернете]

10. http://www.beluch.ru/progr/100comp.htm. [В Интернете]

Предметный указатель