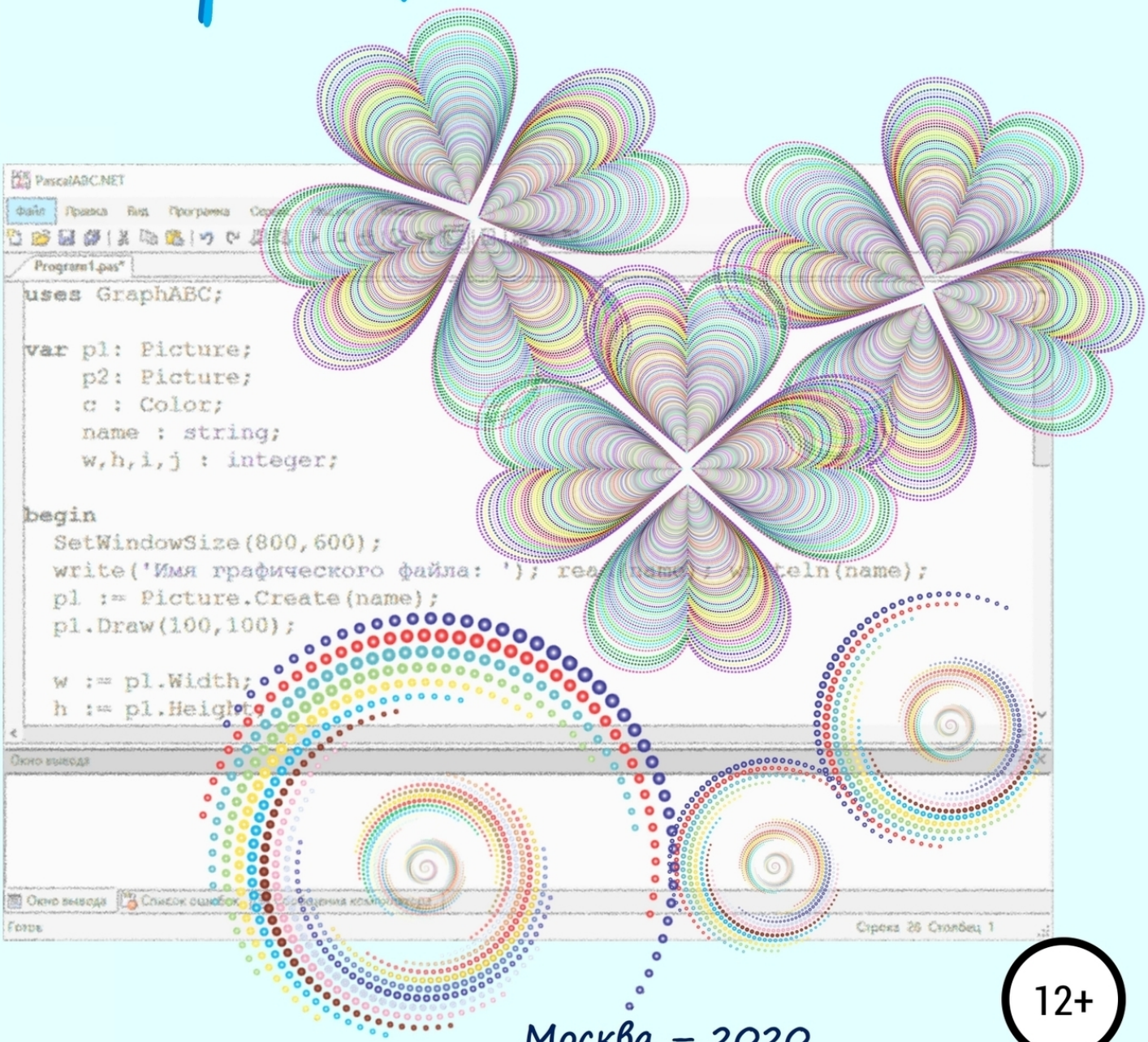
Д. Ю. Усенков

Занимательные задачи по программированию: растровая графика



Москва – 2020

12+

Дмитрий Усенков

Занимательные задачи по программированию обработки растровой графики

«ЛитРес: Самиздат»

2020

Усенков Д. Ю.

Занимательные задачи по программированию обработки растровой графики / Д. Ю. Усенков — «ЛитРес: Самиздат», 2020

ISBN 978-5-532-03190-6

Пособие позволяет на практике познакомиться с основными алгоритмами компьютерной обработки растровой графики, выполняя задания по программированию на языке Visual Basic или Pascal версии ABC.Net. Материал может быть использован для внеурочной учебной деятельности, дополнительного образования, при выполнении учебных проектов и пр.Для школьников, учителей информатики, преподавателей дополнительного образования и для всех, интересующихся принципами работы компьютера с графическими данными.

ISBN 978-5-532-03190-6

Содержание

Введение 5
Visual Basic как лабораторная среда 6
Создание экранной формы 7
Графические возможности Visual Basic 10
PascalABC.Net как лабораторная среда 11
Конец ознакомительного фрагмента. 13

Дмитрий Усенков
Занимательные задачи по программированию обработки растровой графики

Введение

Такая тема, как машинная графика и ее применение, весьма популярна и среди преподавателей информатики, и среди школьников. Однако при программировании задания чаще всего ограничиваются лишь рисованием на экране различных геометрических фигур или составленных из них чуть более сложных картинок. При этом остается в стороне другой, также очень существенный и интересный «пласт» алгоритмов, касающихся графических преобразований растровых изображений – тех, которые мы привыкли выполнять средствами того или иного графического редактора, не задумываясь об их сущности. Это изменение яркости и контраста, преобразование цветных изображений в монохромные, графическая фильтрация и т. д.

Ниже приводится ряд занимательных задач по реализации некоторых наиболее простых графических преобразований, которые помогут понять смысл обычно скрытых от пользователя процессов, происходящих «в недрах» графического редактора при выполнении той или иной команды обработки изображений. Различные вариации этих задач могут быть предложены учащимся для самостоятельного решения на факультативах и олимпиадах по программированию.

Решение таких задач можно рассматривать не только как иллюстрацию принципов работы соответствующих графических преобразований или как задания по программированию, но и как учебный проект по реализации «собственной версии» графического редактора (в том числе коллективный, когда каждый исполнитель проекта реализует какое-то одно графическое преобразование, сложность которого соответствует его уровню подготовки).

При решении предлагаемых в данной брошюре задач (при написании, исполнении и отладке соответствующих программ) предлагается использовать среду визуального программирования Visual Basic (VB) либо транслятор с языка Паскаль – PascalABC.Net. Каждое из этих двух инструментальных средств обладает своими преимуществами и особенностями при работе с растровой графикой, которые будут рассмотрены далее.

Visual Basic как лабораторная среда

Выбор среды визуального программирования Visual Basic (VB) в качестве лабораторной среды обусловлен следующими соображениями.

Во-первых, язык Бейсик, на котором реализуются программные модули в VB, всем хорошо знаком и понятен, а сама оболочка Visual Basic достаточно популярна. (Для написания предлагаемых читателям листингов использована версия Visual Basic 6.0, которая в свое время распространялась по школам, но эти программы могут быть переписаны на новые версии VB или на другом языке программирования, в котором реализован функционал для работы с растровой графикой.)

Во-вторых, концепция *визуального программирования* позволяет легко и быстро «нарисовать» на экране интерфейс простейшего программного приложения, требуемого для испытания разрабатываемых алгоритмов (экранную форму, содержащую исходное изображение, поле для вывода результата и «пусковую» кнопку), сосредоточившись именно на самих алгоритмах графических преобразований.

Наконец, в третьих, Visual Basic предоставляет удобное средство загрузки на экранную форму исходного изображения из любого внешнего файла (например, формата BMP), не требуя разработки соответствующей подпрограммы и обращения к библиотекам работы с растровой графикой.

Методическая ценность использования Visual Basic в качестве лабораторной среды при изучении сложных алгоритмов также существенно повышается благодаря наличию в VB широкого набора встроенных средств отладки программ, что позволяет легко организовать отслеживание работы алгоритмов – в частности, контроль значений выбранных переменных. При этом режим отладки позволяет выполнять программу «по шагам» (по одной строке листинга при каждом нажатии клавиши **F8**), просматривая в отдельном окне выведенные системой отладки текущие значения отслеживаемых переменных.

Напомним, что управление средствами отладки сосредоточено в меню **Debug** (**Отладка**):

– пункт **Add Watch** (**Добавить Наблюдателя**) позволяет добавить любую из имеющихся в листинге переменных в список отслеживания значений (можно предварительно выделить в листинге имя требуемой переменной, тогда оно автоматически будет занесено в соответствующую графу окна добавления переменной, даже если выделенное имя не было скопировано в буфер обмена);

– пункт **Edit Watch (Правка Наблюдателя)** раскрывает окно управления списком отслеживаемых переменных, где можно добавлять новые переменные или удалять существующие, ставшие ненужными;

– пункт **Toggle Breakpoint** (**Включить Финиш-Точки**) либо горячая клавиша **F9** позволяют создавать (или, при повторном вызове, снимать) в листинге контрольные точки останова. При вызове этого пункта или нажатии клавиши **F9** текущая строка листинга, на которой находился текстовый курсор, выделяется бордовой фоновой подсветкой, а запущенная на выполнение программа, дойдя до отмеченной строки, приостанавливается, и Visual Basic переходит в режим отладки.

Запуск написанных программ и просмотр получаемых результатов мы будем производить средствами оболочки VB, хотя ничто не запрещает при желании и оттранслировать результаты работы в виде полноценного Windows-приложения.

Создание экранной формы

После запуска среды Visual Basic на экране появляется ряд отдельных окон и панелей (рис. 1), среди которых для нас важными являются:

– основная панель вверху экрана (главное меню и строка кнопок);

– панель инструментов (слева), содержащая кнопки для добавления на создаваемую экранную форму тех или иных объектов интерфейса;

– сама экранная форма – при открытии VB создается по умолчанию, для добавления новой экранной формы нужно воспользоваться кнопкой

– окно проекта (справа вверху), содержащее перечень созданных экранных форм и стандартных библиотек (если созданная экранная форма отсутствует на экране, ее окно можно раскрыть двойным щелчком мыши на соответствующей строке окна проекта);

– окно свойств объекта – раскрывается автоматически; при закрытии может быть повторно раскрыто с помощью кнопки

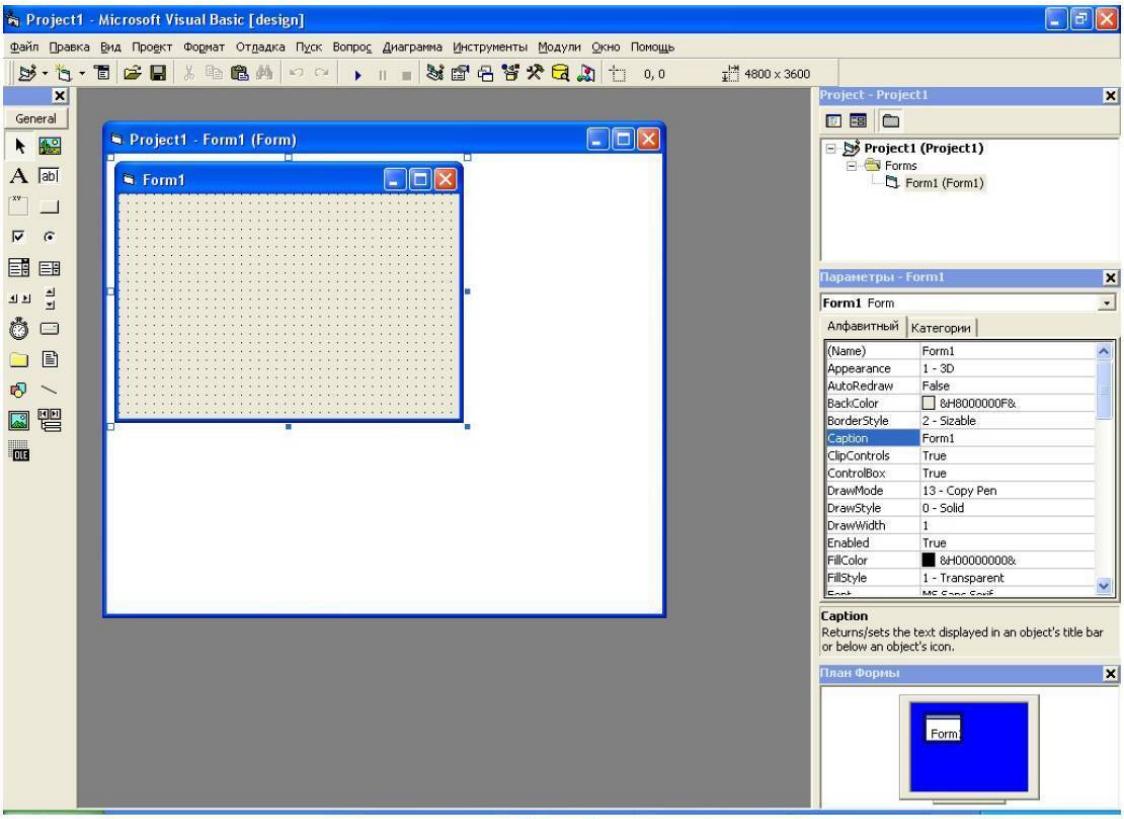


Рис. 1

Подготовим простейшую экранную форму, с помощью которой будет производиться проверочный запуск программ, созданных в качестве решения задач.

1. Перетаскивая мышью углы и границы окна, увеличим размеры предлагаемой экранной формы до требуемых.

2. Выбирая в панели инструментов кнопку

разместим («нарисуем») на экранной форме два объекта «поле рисунка» (**Picture Box**), расположив их рядом по горизонтали.

3. Выбрав в панели инструментов кнопку

разместим в нижней части экранной формы управляющую кнопку (**Command Button**).

4. Выполнив однократный щелчок мышью на каждом из добавленных объектов и обратившись к окну свойств объекта, зададим следующие значения их параметров:

Параметр	Значение	Описание
Первый объект `PictureBox` *(левый)*		
Name	**Picture1**	имя объекта, используемое в листинге программ
AutoSize	**True**	разрешить автомасштабирование поля по размерам загруженного рисунка
ScaleMode	**3 - Pixel**	измерение размеров в пикселях
Второй объект `PictureBox` *(правый)*		
Name	**Picture2**	имя объекта, используемое в листинге программ
ScaleMode	**3 - Pixel**	измерение размеров в пикселях
Объект `Command Button`		
Name	**Command1**	имя объекта, используемое в листинге программ
Title	**СТАРТ**	надпись на кнопке

Остальные «оформительские» параметры для кнопки и экранной формы в целом (цвет фона, цвет надписи на кнопке и пр.) можно установить по своему желанию.

5. Подготовим с помощью любого графического редактора несколько исходных картинок формата BMP (16-битный или 256-цветный режим) с размерами не более 300300 пикселей по ширине и высоте для проверки работы создаваемых алгоритмов. Желательно подготовить хотя

бы по одной цветной и полутоновой (оттенки серого) картинке с растровой фотографией (либо рисунком с фотографическим качеством) и с «плакатной графикой» (рисунок, содержащий достаточно большие области с гладкой закраской). Примеры таких картинок показаны на рис. 2.

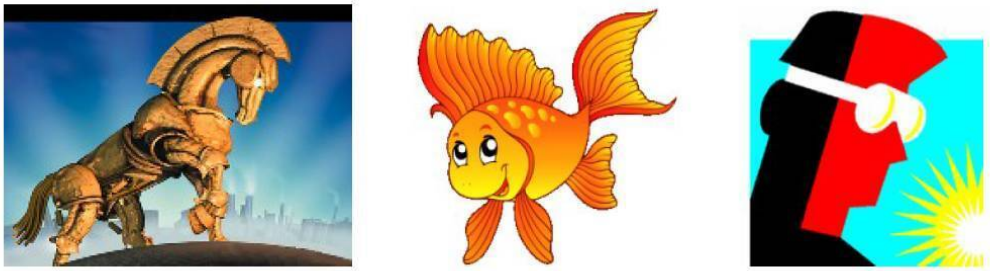


Рис. 2

6. Для объекта **Picture1** (левое поле рисунка) в окне свойств выберем строку **Picture** и щелкнем мышью на ней, а затем на появившейся в правом верхнем углу окна свойств (в строке ввода значений параметров) кнопке  и в выведенном на экране стандартном окне открытия файлов выберем желаемый графический файл. Щелкнем мышью на кнопке **ОК**, – выбранный рисунок будет помещен в левое поле, которое автоматически примет требуемые размеры.

Аналогичным способом можно менять рисунок, загружаемый в левое поле перед запуском программ, реализующих те или иные графические преобразования.

Полученный возможный внешний вид созданной экранной формы показан на рис. 3.

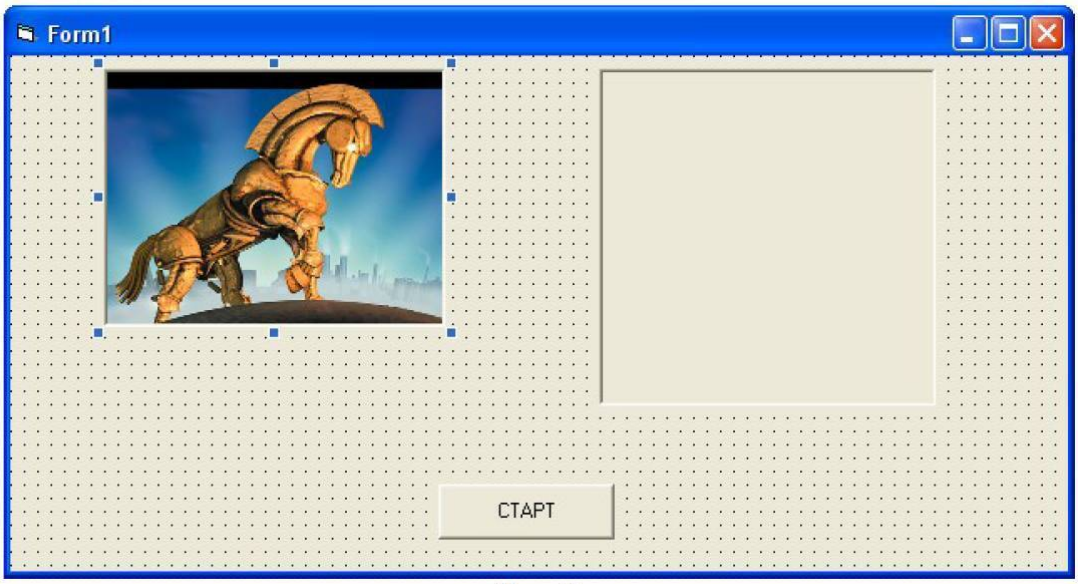


Рис. 3

Графические возможности Visual Basic

Среда Visual Basic предоставляет весь типовой набор стандартных графических операторов, который присущ современным языкам программирования высокого уровня, в том числе рисование отрезков, прямоугольников и окружностей, управление толщиной и стилем рисуемых линий, закраской и пр. Но нас будут интересовать только операции для работы с отдельными пикселями и кодированием цвета:

Point(x,y) – возвращает значение цвета точки с координатами (*x*,*y*), закодированное в виде 2-байтного числа;

PSet (x,y),c – выводит точку с координатами (*x*,*y*) и цветом, указанным в виде 2-байтного числа;

RGB(R,G,B) – возвращает 2-байтное значение цвета, синтезируя его из отдельных 1-байтовых значений яркостей основных цветов – **R** (красный), **G** (зеленый) и **B** (синий), лежащих в диапазоне от 0 до 255. Обратная функция для разбиения 2-байтного значения цвета на значения яркостей основных цветов, в VB, к сожалению, отсутствует. О том, как заменить ее в программе, будет рассказано чуть позже.

Следует заметить, что в VB работа с пикселями реализуется в поле рисунка (**Picture Box**), а имя соответствующего объекта (значение параметра **Name**) указывается через точку при вызове функций **Point** и **PSet**, например:

```
PixColor = Picture1.Point(x, y)
```

PascalABC.Net как лабораторная среда

Транслятор с языка программирования Паскаль – PascalABC.Net – не является средой визуального программирования, поэтому просто «нарисовать» на экране какую-либо экранную форму для создания приложения в нем не удастся. Имеющиеся в составе PascalABC.Net стандартные библиотеки позволяют лишь открыть на экране одно-единственное окно заданных размеров (в нашем случае – аналогичное окну экранной формы в VB), в котором производится как ввод/вывод текста (при помощи привычных read и write либо readln и writeln), так и вывод растровой графики в указанных в программе местах окна (рис. 4).

Однако значительным преимуществом PascalABC.Net является его общедоступность и бесплатность. В отличие от коммерческого Visual Basic, транслятор PascalABC.Net является свободно распространяемым (http://pascalabc.net), обеспечивает поддержку всех современных версий Windows и прост в освоении и использовании, а язык Паскаль во многих школах является «базовым» при преподавании информатики.

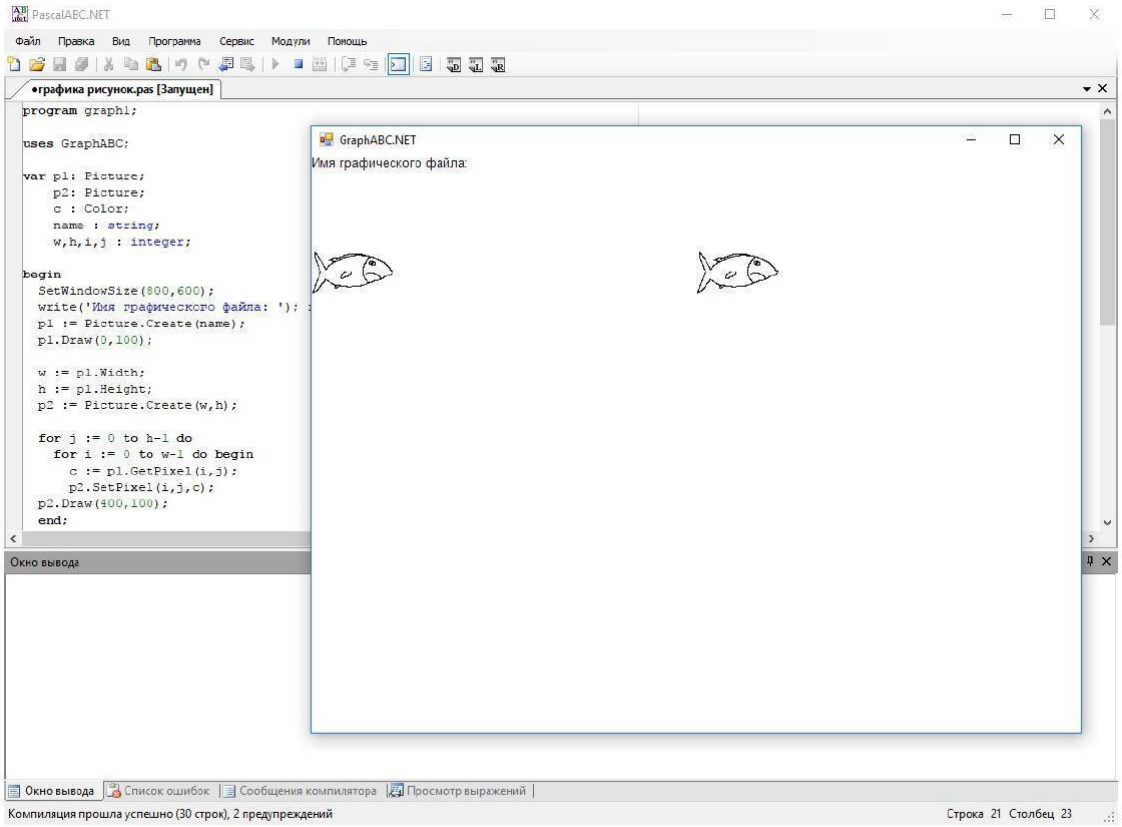


Рис. 4

Транслятор PascalABC.Net также представляет возможности отладки программ:
– кнопка

панели инструментов либо клавиша **F8** позволяет выполнять программу «по шагам» (каждое нажатие указанной кнопки или клавиши исполняет одну очередную строку программы, которая выделяется желтой подсветкой);

– щелчки мышью на левом поле окна программы напротив соответствующих ее строк позволяют устанавливать на этих строках контрольные точки останова (красный кружок; соответствующая строка также выделяется красной подсветкой) либо повторным щелчком на ранее установленных контрольных точках снимать их. После этого запуск программы на исполнение вызывает выполнение всех строк до установленной контрольной точки, а затем транслятор останавливает программу и переходит в «пошаговый» режим;

– имеется возможность просмотра в процессе пошагового исполнения программы значений выбранных переменных или заданных выражений. Все используемые в программе переменные и массивы доступны для просмотра их значений на вкладке **Локальные переменные**

Конец ознакомительного фрагмента.