Информатика

Абрамян Михаил Эдуардович
Кандидат физико-математических наук, доцент кафедры алгебры и дискретной математики Южного федерального университета, руководитель проекта Programming Taskbook.

Михалкович Станислав Станиславович
Кандидат физико-математических наук, доцент кафедры алгебры и дискретной математики Южного федерального университета, руководитель проекта PascalABC.NET.

PascalABC.NET для школьников

Язык программирования Паскаль был создан швейцарским ученым Никлаусом Виртом в 1970 г. как язык для обучения программированию. С тех пор появилось множество версий языка Паскаль. Некоторые из них оказались очень удачными и получили широкое распространение. Пожалуй, наиболее известными являются системы программирования Turbo Pascal для DOS и Delphi для Windows, а также их бесплатные аналоги – Free Pascal и Lazarus.

Предлагаемая статья посвящена системе программирования PascalABC.NET (ABC по-английски – «азбука», «основы»), созданной на факультете математики, механики и компьютерных наук Южного федерального университета.

Первая версия системы (она называлась Pascal ABC) появилась в 2002 году и была ориентирована исключительно на школьников. В настоящее время система PascalABC.NET базируется на платформе Microsoft .NET Framework и может использоваться для профессионального программирования. Однако её главная задача остаётся прежней: помогать школьникам и студентам учиться программированию. Мы рассмотрим возможности системы PascalABC.NET, проведя несколько уроков.

Система PascalABC.NET «с высоты птичьего полета»

Система PascalABC.NET, как и любая система программирования, содержит много различных компонентов.

Основным компонентом любой системы является её *язык программирования*. Язык PascalABC.NET имеет очень богатые и разнообразные возможности. Вы можете использовать базовые средства, которые имеются в любой реализации «традиционного» Паскаля (и с которых начинается обучение программированию). Однако вам доступны и многочисленные расширения «традиционного» Паскаля. Многие из этих расширений первоначально появились в языке Delphi, другие связаны с платформой .NET Framework. Язык PascalABC.NET содержит такие элементы, как классы, исключения, обобщения, интерфейсы, делегаты. С его помощью можно разрабатывать программы, используя различные *технологии программирования*: объектно-ориентированную, событийно-ориентированную, технологию параллельного программирования. Все эти возможности вы можете изучить, не выходя за пределы системы PascalABC.NET, а полученные знания позволят вам быстро освоить и другие современные языки программирования, такие как C# и Java.

Другим важным элементом любой системы программирования является *среда разработки*. Это программа, находясь в которой, вы сможете выполнять все действия по разработке своих программ – от набора их текстов до запуска на выполнение. Именно благодаря наличию среды разработки мы можем очень быстро писать новые программы и (что ещё важнее!) быстро находить в этих программах ошибки и исправлять их. К сожалению, среды разработки большинства современных систем программирования (например, Delphi или Lazarus) являются очень сложными и поэтому не слишком хорошо подходят для использования на начальных этапах обучения. Поскольку система PascalABC.NET создавалась для обучения, её среда имеет простой русскоязычный интерфейс. Однако в ней скрыто много возможностей, которые легко использовать. На рисунке 1 изображено окно среды PascalABC.NET во время типичного сеанса работы.

В среде можно одновременно разрабатывать несколько программ, причём встроенный в среду редактор имеет различные средства, облегчающие ввод текста программы: подсветку синтаксиса, контекстно-зависимые подсказки, систему автоматического форматирования. Для ввода исходных данных и вывода результатов предназначено специальное *окно вывода* в нижней части основного окна. В этом же разделе основного окна отображаются сообщения об ошибках. Все сообщения и подсказки выводятся на русском языке.

В состав любой системы программирования входит также набор стандартных библиотек (модулей), расширяющих возможности языка. Система PascalABC.NET включает набор библиотек, аналогичных библиотекам системы Delphi. Кроме того, в ней можно также использовать весь богатейший арсенал стандартных библиотек платформы .NET.

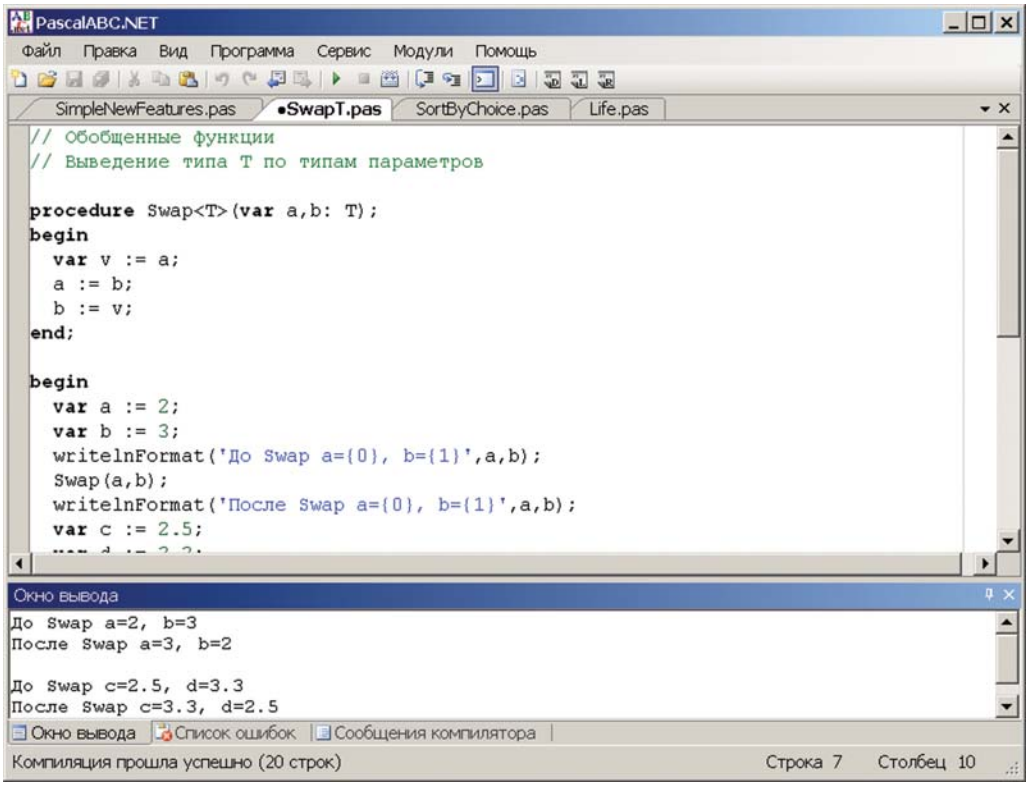

Рис. 1. Окно среды PascalABC.NET

Являясь учебной системой программирования, PascalABC.NET содержит в своём составе ряд специальных *обучающих модулей*. В их числе модули *исполнителей* Робот и Чертёжник, широко используемые в школьной информатике, модули растровой и векторной графики GraphABC и ABCObjects, модуль Forms для построения простых оконных приложений и т. д.

Замечательной возможностью PascalABC.NET являются так называемые *проверяемые задания* – задания, автоматически генерируемые и проверяемые системой PascalABC.NET. Большое число проверяемых заданий по различным разделам программирования содержит электронный задачник Programming Taskbook, входящий в состав системы.

Урок 1. Исполнитель Робот

Исполнители широко используются для быстрого обучения конструкциям языка. Система PascalABC.NET позволяет работать с исполнителями Робот и Чертёжник, которые «живут» на клеточном поле. Эти исполнители реализованы в специальных модулях Robot и Drawman.

Мы рассмотрим исполнителя Робот. Этот исполнитель имеет простую систему команд: Left, Right, Up, Down для перемещения в соответствующем направлении,

Paint для закрашивания текущей клетки, а также логические команды WallFromLeft, WallFromRight, WallFromUp, WallFromDown, FreeFromLeft, FreeFromRight, FreeFromUp, FreeFromDown для проверки, есть ли стена в соответствующем направлении рядом с Роботом или это направление является свободным.

Модуль Robot содержит не только команды для Робота. В него встроено большое число заданий, связанных с различными темами начального курса программирования. Система PascalABC.NET может самостоятельно проверить правильность выполнения любого задания и при его правильном выполнении поздравить вас с успехом.

Чтобы ускорить выполнение заданий, можно воспользоваться специальным средством среды PascalABC.NET – мастером по загрузке заданий. Для вызова этого мастера нажмём кнопку  на панели инструментов. На экране появится окно мастера. Набрав в нем текст «RB», мы увидим в окне список всех наборов заданий, связанных с исполнителем Робот (рис. 2).

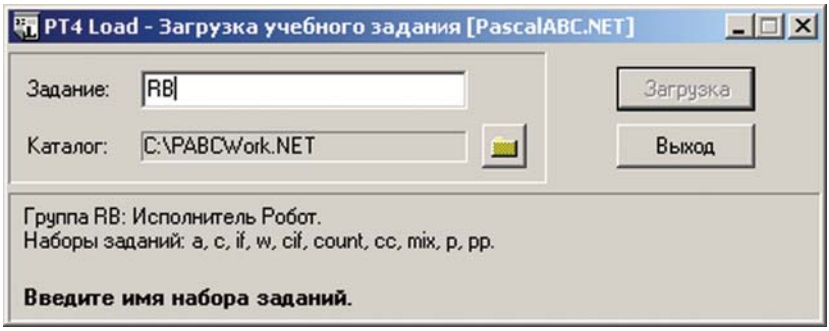

Рис. 2. Окно мастера по загрузке проверяемых заданий

Выберем задание номер 3 из группы pp (имя задания примет вид «RBpp3») и нажмём кнопку **Загрузка**. В результате в рабочем каталоге (по умолчанию это каталог C:\PABCWork.NET) будет создан файл с именем RBpp3.pas, содержащий заготовку для выбранного задания:

```
uses Robot;
begin
  Task('pp3');
end.
```

Директива **uses** подключает к программе модуль Robot, а процедура `Task('pp3')` обеспечивает инициализацию выбранного задания pp3.

Запустив эту программу (для этого достаточно нажать клавишу [F9]), мы увидим на экране окно, связанное с исполнителем Робот (рис. 3). Наше задание состоит в том, чтобы провести Робота по спиральному коридору в центральную клетку. Чтобы закрыть окно Робота, нажмём клавишу [Esc].

Нетрудно заметить, что для решения задачи Робот должен пройти четыре витка спирали (в направлении «вниз–вправо–вверх–влево»), после чего опуститься вниз на одну клетку. Чтобы не удариться о стену, можно использовать логические команды Робота. В результате получаем следующее решение:

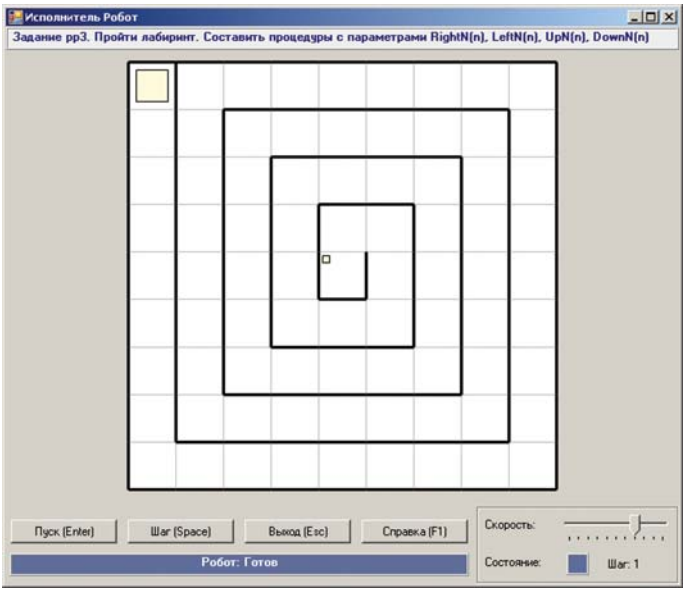

Рис. 3. Окно исполнителя Робот

```
uses Robot;
var i: integer;
begin
  Task('pp3');
  for i := 1 to 4 do
  begin
    while FreeFromDown do
      Down;
    while FreeFromRight do
      Right;
    while FreeFromUp do
      Up;
    while FreeFromLeft do
      Left;
  end;
  Down;
end.
```

После запуска этого варианта программы мы должны активизировать Робота, нажав кнопку **Пуск** в его окне (или клавишу [Enter]). Когда при выполнении программы Робот дойдёт до центральной клетки и остановится на ней, в строке состояния окна Робота появится сообщение о том, что задание выполнено.

Приведённое решение не является единственно возможным. Например, вместо использования логических команд можно явным образом указать число шагов в каждом из четырёх направлений (это число будет зависеть от номера витка спирали). Можно описать и использовать при решении задачи вспомогательные процедуры, позволяющие Роботу перемещаться в указанном направлении до стены. Можно также использовать вспомогательные процедуры, позволяющие выполнить Роботу указанное число шагов в заданном направлении.

Урок 2. Новые возможности языка

PascalABC.NET предлагает несколько расширений традиционного синтаксиса языка Паскаль. Эти расширения делают программы короче и нагляднее и позволяют избежать многих ошибок, которые часто совершают начинающие программисты.

1. Переменные можно описывать не только в начале программы в

специальном разделе описаний, но и в разделе операторов – непосредственно перед тем фрагментом программы, где вы собираетесь использовать эти переменные.

2. При описании переменной ей можно присваивать значение. Например:

```
var p: integer := 1;
```

Используя эту возможность, вы реже будете забывать инициализировать переменные.

3. Тип переменной может быть определён автоматически по типу выражения, которое ей присваивается при описании. Например:

```
var p := 1;
```

Поскольку 1 – константа целого типа, переменная p также будет иметь целый тип.

4. Параметры цикла можно описывать прямо в его заголовке. Здесь тоже можно пользоваться автоопределением типа:

```
for var i := 1 to 10 do
  Write(i);
```

Эта возможность не позволяет использовать переменную i вне цикла. Так, следующий код вызовет ошибку компиляции (из-за чего она произошла?):

```
for var i := 1 to 10 do;
  Write(i);
```

5. Из языков, подобных языку C, взяты комбинированные операторы присваивания вида +=, *= и т. д. Например, оператор i += 2 является сокращенным обозначением двух действий – сложения и присваивания: `i := i + 2`. Комбинированные операторы присваивания более удобны для восприятия и прочтения; например, оператор += можно прочесть как «увеличить на», а оператор *= – как «увеличить в».

6. Для удобства ввода одного значения соответствующего типа определены функции `ReadInteger`, `ReadReal`, `ReadString` и т. д. Например, для описания переменной n и немедленного ввода её значения с клавиатуры можно использовать единственный оператор:

```
var n := ReadInteger;
```

Проиллюстрируем все описанные выше возможности на примере простой программы. Эта программа вычисляет сумму десяти вещественных чисел, вводимых с клавиатуры:

```
begin
  var s := 0.0;
  Writeln('Введите десять',
    ' чисел:');
  for var i := 1 to 10 do
    s += ReadReal;
  Writeln('Сумма чисел',
    ' равна ', s);
end.
```

Урок 3. Электронный задачник

Если вы набрали и запустили на выполнение программу, приведённую в конце предыдущего урока, то для проверки её работы вам будет необходимо ввести десять исходных чисел, после чего программа выведет их сумму, правильность которой вам придётся проверить «вручную». Нельзя ли «переложить» на саму систему программирования такие действия, как подготовка и ввод исходных данных и, главное, – проверка полученных результатов? Это позволило бы нам гораздо быстрее тестировать наши программы, выявлять и исправлять ошибки и, в конечном счёте, решать за то же время больше задач. Система PascalABC.NET может и это, поскольку в неё встроен электронный

задачник по программированию Programming Taskbook.

Давайте посмотрим, как будет выглядеть процесс решения задачи о нахождении суммы десяти чисел с использованием задачника. Эта задача является первой в группе заданий Series, посвящённой обработке числовых последовательностей. Вызовем ещё раз мастера по загрузке проверяемых заданий, введём имя задания «Series1» и нажмём кнопку **Загрузка**.

Полученная заготовка программы будет очень похожа на заготовку проверяемого задания для Робота (отличается лишь имя подключаемого модуля):

```
uses PT4;
begin
  Task('Series1');
end.
```

Запустив эту программу на выполнение, мы увидим на экране окно задачника с выбранным заданием (рис. 4).

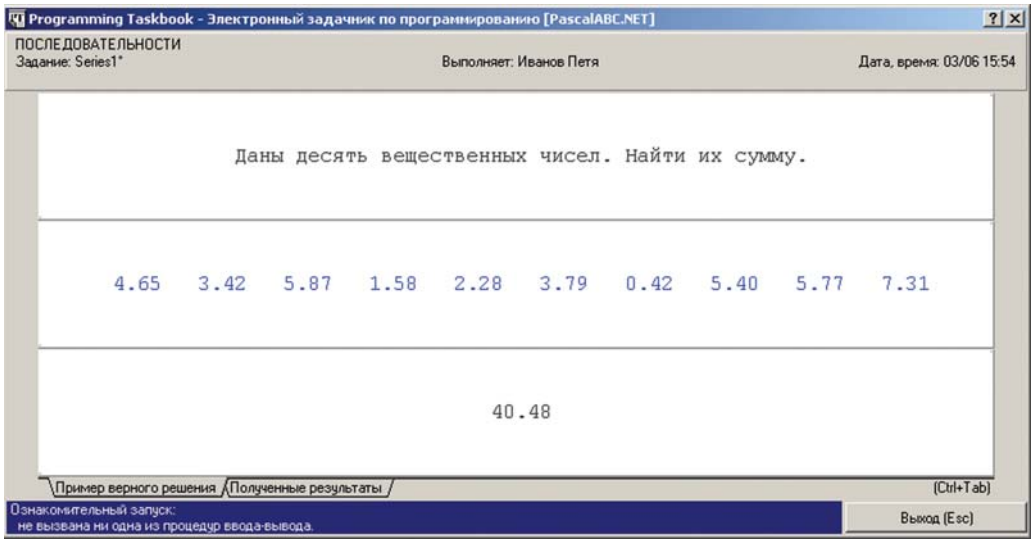

Рис. 4. Окно электронного задачника

В этом окне приводится формулировка задания, пример набора исходных данных и пример правильного решения для этих исходных данных. В нижней части окна сообщается, что текущий запуск программы считается ознакомительным. Действительно, мы не вводили и не выводили данные, поэтому проверять задачнику нечего.

Начнём выполнять задание. Чтобы не дублировать приведённое выше решение, будем пользоваться только «традиционными» средствами Паскаля. Первый вариант решения будет таким:

```
uses PT4;
var
  i: integer;
  x, s: real;
begin
  Task('Series4');
  s := 0;
  for i := 1 to 10 do
  begin
    Read(x);
    s := s + x;
  end;
end.
```

Вы, конечно, нашли ошибку в этом решении: мы не выводим на экран результаты работы програм-

мы. Кроме того, мы не выводим никаких приглашений к вводу исходных данных. Но отсутствие приглашений как раз не является ошибкой – ведь нам не придётся набирать исходные данные на клавиатуре.

Как отреагирует задачник на такой вариант решения? Разумеется, он выдаст сообщение о том, что требуемый результат не выведен на экран. Исправим программу, добавив в её конец оператор вывода (пояснения к выводимым данным добавлять не следует, поскольку при необходимости это сделает сам задачник):

```
Write(s);
```

При запуске нового варианта программы задачник выведет сообщение: «*Верное решение. Тест номер 1 (из 5)*». Это значит, что одно тестовое испытание нашей программы прошло успешно. Но, как мы знаем, для того чтобы убедиться в правильности программы, её надо протестировать на *разных* наборах исходных данных. Задачник тоже это знает, поэтому после первого успешного запуска он ещё не считает задание выполненным. Для этого требуется несколько успешных запусков (в данном случае пять), проведённых подряд, после чего задачник выведет сообщение «*Задание выполнено!*».

Урок 4. Графические модули

В PascalABC.NET имеются два стандартных графических модуля: `GraphABC` и `ABCObjects`. Первый из них содержит все необходимые средства для создания растровых изображений, второй позволяет работать с графическими объектами, управляя их свойствами (тем самым процесс создания изображений становится похож на рисование в векторном редакторе). В программе можно совместно использовать оба этих модуля.

Графический модуль позволяет создавать изображение в графическом окне, состоящем из *пикселов* – точек, которые можно окрашивать в различные цвета. Пикселы нумеруются слева направо и сверху вниз, начиная с нуля. Таким образом, левый верхний пиксел имеет координаты (0, 0), а ось *OY* считается направленной вниз. Графическое окно можно настраивать с помощью объекта `Window`, используя такие его свойства как `Width` (ширина), `Height` (высота) и `Title` (заголовок). Для обращения к свойствам используется *точечная нотация*, например, `Window.Title`.

Для работы с цветами в `GraphABC` служит тип `Color`. В нём содержится более полусотни стандартных цветов, для обращения к которым также следует использовать точечную нотацию: `Color.Green`, `Color.Red` и т. д. Кроме того, с помощью функции `RGB(r,g,b)` можно сформировать любой цвет с заданными интенсивностями красной, зелёной и синей составляющих.

Для рисования в графическом окне используются так называемые *графические примитивы*. К их числу относятся: `Line` (отрезок), `Rectangle` (прямоугольник), `Ellipse` (эллипс) и т. д. Рамки примитивов рисуются *пером* `Pen`, у которого есть такие свойства как `Color` (цвет) и `Width` (ширина), а внутренность замкнутых примитивов рисуется *кистью* `Brush`, у которой тоже имеется свойство `Color`. После установки свойств пера и кисти все последующие графические примитивы рисуются с этими свойствами.

Рассмотрим простую программу, иллюстрирующую большинство из описанных возможностей.

```
uses GraphABC;
begin
  Window.Title :='Фигура';
  Window.Width := 200;
  Window.Height := 200;
  Pen.Width := 3;
  Brush.Color :=
    Color.Yellow;
  Rectangle(50, 50, 150,
    150);
  Line(50, 150, 150, 50);
  Line(50, 50, 150, 150);
end.
```

В результате выполнения программы графическое окно примет вид, приведённый на рисунке 5.

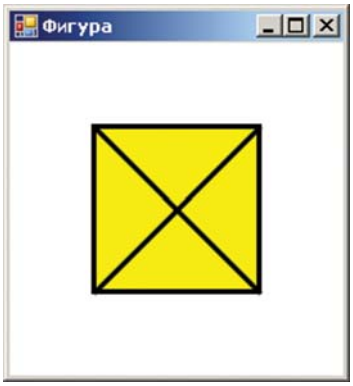

Рис. 5. Пример рисунка в графическом окне

Графические модули можно использовать для наглядного представления данных. В качестве примера используем модуль ABCObjects для визуализации стандартного алгоритма поиска минимального и максимального элементов в массиве.

Будем изображать массив в виде *гистограммы*: каждый элемент массива будет прямоугольником с высотой, равной значению этого элемента. Нам потребуются возможности обоих модулей, поэтому заготовка программы будет иметь следующий вид:

```
uses GraphABC, ABCObjects;
begin
end.
```

Все операторы программы будем указывать между словами begin и end. Раздел описаний останется пустым, так как все переменные будут описываться непосредственно перед их использованием.

Для вывода на экран прямоугольников в модуле ABCObjects предусмотрен класс RectangleABC. Чтобы нарисовать прямоугольник, достаточно вызвать для него *конструктор* Create, указав в качестве параметров свойства этого прямоугольника: положение в окне (координаты левого верхнего угла), ширину, высоту и цвет заливки. Таким образом, для изображения гистограммы достаточно в цикле создать все 10 прямоугольников (их высота выбирается случайным образом из диапазона значений 20–100; для этого используется функция Random (20, 100)):

```
Window.Title := 'Нахождение минимума и максимума';
Window.Height := 120;
Window.Width := 310;
var a: array [1..10] of RectangleABC;
for var i := 1 to 10 do
  a[i] := RectangleABC.Create(i*30 - 20, 10, 20,
    Random(20, 100), Color.White);
```

Чтобы хранить информацию о минимальном и максимальном элементах массива, будем использовать переменные nmin и nmax, за-

писывая в них номера минимального и максимального элементов массива из уже проанализированных:

```
var nmin := 1;
var nmax := 1;
for var i := 2 to 10 do
  if a[i].Height < a[nmin].Height then
    nmin := i
  else
    if a[i].Height > a[nmax].Height then
      nmax := i;
```

Каким образом выделить на нашей гистограмме минимальный и максимальный прямоугольники? Здесь нам поможет то обстоятельство, что все нарисованные прямоугольники являются *объектами*, и достаточно изменить какое-либо их свойство (например, цвет), чтобы это изменение немедленно сказалось на их изображении. Закрасим минимальный прямоугольник красным цветом, а максимальный – синим:

```
a[nmin].Color:= Color.Red;
a[nmax].Color:= Color.Blue;
```

Результат работы программы приведён на рисунке 6.

Рис. 6. Гистограмма с выделенными элементами

Заключение

Мы познакомились лишь с небольшой частью возможностей системы PascalABC.NET. Наш рассказ не затронул такие темы, как программы, управляемые событиями, оконные приложения, прикладные классы .NET. Мы не рассмотрели проверяемые задания для исполнителя Чертёжник, задачи на массивы, строки, файлы, рекурсивные алгоритмы и динамические структуры данных, входящие в состав задачника Programming Taskbook. Мы не описали веб-среду PascalABC.NET WDE, которая позволяет разрабатывать программы и запускать их на выполнение непосредственно в окне веб-браузера.

Если вы захотели больше узнать о системе PascalABC.NET, посетите её сайт, название которого легко запомнить: **pascalabc.net**. С этого сайта вы можете установить PascalABC.NET на ваш компьютер. Система PascalABC.NET не требует специальной регистрации, поэтому все её возможности будут доступны сразу после установки.