Лицензия

Введение

Язык программирования PascalABC.NET – мощный современный язык, включающий в себя классический Паскаль, большинство возможностей языка Object Pascal среды Delphi и многочисленные собственные расширения. Компилятор и библиотеки языка PascalABC.NET свободно распространяются по лицензии LGPL v3.

На базе языка PascalABC.NET создана одноименная интегрированная среда разработки и отладки программ, поддерживающая технологию IntelliSence. Реализация выполнена на платформе Microsoft .NET Framework и опирается на использование ее библиотек. В операционных системах Linux и MacOS требуется среда проекта Mono, под которой запускается консольный компилятор.

PascalABC.NET – мультипарадигменный язык. Он позволяет программировать в структурном, объектно-программированном и функциональном стиле, а также смешивать эти стили в одной программе. Насколько это было возможно, PascalABC.NET впитал лучшие идеи, реализованные в языках C#, Python и Haskell. Большое количество «синтаксического сахара» делает программирование на PascalABC.NET простым и комфортным как для начинающих, так и для имеющих опыт в создании программ.

Немного истории

Первый высококачественный компилятор языка Паскаль для персональных компьютеров с операционной системой DOS появился в 1983 году под названием Turbo Pascal. В дальнейшем он был переименован в Borland Pascal и под таким именем существует до сих пор.

Популярность Turbo/Borland Pascal привела к выпуску множества литературы, посвященной программированию в этой, замечательной для своего времени среде. Язык массово изучался в школах и вузах. В 1994 году компания Borland выпустила последнюю версию 7.1 и затем отказалась от дальнейшей поддержки проекта в связи переходом к Borland Delphi. Компания не открыла исходные тексты, оставив лицензию проприетарной. Это привело к тому, что что любой экземпляр Turbo/Borland Pascal, установленный на компьютер в последние два десятка лет, оказывается нелицензионной («пиратской») копией. Об этом не следует забывать учителям информатики школ и других учебных заведений, до сих пор предлагающим учащимся «скачать себе где-нибудь Турбо Паскаль».

Появление Borland Delphi несколько снизило интерес к Borland Pascal. Визуальная среда быстрой разработки, поддержка объектно-ориентированных технологий – все это вызывало большой интерес, сдерживаемый лишь ценой на продукт. Использовать Delphi для целей обучения могли себе позволить лишь немногие вузы.

Шли годы, система совершенствовалась, но цена на нее особенно не снижалась. Параллельно развивалась операционная система Windows, для которой Borland Pascal был чужеродным – он работал в DOS-режиме. Windows XP оказалась последней операционной системой, в которой еще можно было беспроблемно использовать Borland Pascal.

В 1997 стартовал проект по свободно распространяемой мультиплатформенной версии Паскаля, получивший название Free Pascal (FPC) - реализация на базе языка Object Pascal. Существует также открытый проект визуальной разработки программ Lazarus, основанный на FPC.

Языки программирования развиваются непрерывно. Здесь Паскалю не повезло. Заложенное еще его создателем, Н. Виртом, стремление сделать язык простым, породило множество ограничений, сдерживающих развитие Паскаля. Язык оказался неспособным поддерживать современные идеи, методы и технологии программирования. Как следствие, в коммерческой сфере интерес к Паскалю со временем пропал, и язык остался востребованным лишь в сфере образования.

В 2003 году энтузиастами Южного федерального университета (ЮФУ, г. Ростов-на-Дону) была создана учебная среда программирования Pascal ABC (ABC в английском языке имеет значение «азбука»), как альтернатива платным средам на базе языка Паскаль. Pascal ABC очень быстро набрал популярность в учебных заведениях России и стран бывшего СССР. Впоследствии разработчики решили дать Паскалю вторую жизнь. В процессе выработки спецификаций «нового Паскаля» была учтена потребность обеспечить быстрый последующий переход к изучению языка C#, а также шуточные пожелания преподавателей ЮФУ «писать в одну-две строчки» любые программы, которые даются в школах. В 2009 году появилась первая стабильная версия открытого проекта PascalABC.NET 1.2.

За прошедшие годы произошли существенные изменения в коллективе разработчиков, язык также претерпел множество изменений. На время написания этой книги актуальной является версия PascalABC.NET 3.4.

В книге все реализации языка Паскаль, кроме PascalABC.NET, будут называться общим термином «***базовый Паскаль***». В случае необходимости может указываться конкретная реализация.

Зачем изучать PascalABC.NET?

1. Базовый Паскаль за пределами сферы образования практически не используется. Можно возразить, что и PascalABC.NET также не используется за пределами сферы образования. Но если изучать Паскаль, то такой, на котором удобно и быстро писать и отлаживать программы.

2. PascalABC.NET позволяет в очень короткий срок перейти к изучению современных языков программирования, например, C#.

3. PascalABC.NET позволяет изучить современные технологии программирования, научиться оперировать последовательностями и кортежами, писать программы с элементами функционального стиля, применять стандартные коллекции.

4. PascalABC.NET дает возможность отказаться от концепции статических массивов в пользу динамических, существенно упрощая работу с массивами, в том числе, при обмене с процедурами и функциями.

5. Возможность решать в несколько строчек большинство задач, даваемых в учебниках по информатике, позволяет преподавателю сосредоточиться на алгоритмах решения, а не переписывать каждый раз одни и те же фрагменты реализации. Быстрое написание и отладка позволяют за одно занятие рассмотреть намного больше задач, повышая эффективность обучения.

6. Алгоритм, реализованный на PascalABC.NET нагляден и легко читается, делая лишним рисование блок-схем. Вносить изменения в готовую программу также быстро и легко. Дополнительным удобством является возможность использовать в программах идентификаторы, содержащие буквенные символы, отличные от латиницы (кириллические, греческие и т.д.).

7. Скорость выполнения готовой программы в большинстве случаев такая же, как у программы, написанной на C#.

8. В PascalABC.NET всегда остается возможность рассмотреть подробную реализацию нужного алгоритма на «низкоуровневом» базовом Паскале.

9. В PascalABC.NET можно обращаться к любым библиотекам платформы .NET Framework. Можно создавать свои библиотеки, которые затем использовать, в том числе, и в программах. написанных на других .NET-языках.

О требовании эффективности программ

При знакомстве с заданиями по информатике, предлагаемыми в процессе обучения школьников, учащихся и студентов, встречается пожелание, а то требование написать программу, «эффективную по памяти и/или эффективную по времени». Вот, например, подобные критерии ФИПИ, предлагаемые для оценивания результатов сдачи ЕГЭ-2019: «Программа считается эффективной по времени, если при увеличении количества исходных чисел N в k раз время работы программы увеличивается не более чем в k раз. Программа считается эффективной по памяти, если память, необходимая для хранения всех переменных программы, не превышает 1 килобайта и не увеличивается с ростом N».

Смысл такого требования для экзаменационной или иной проверочной работы ясен: оценка навыков алгоритмического мышления. Но к программированию на конкретном языке это имеет очень отдаленное отношение.

ЭВМ трех первых поколений занимали огромные площади, требовали больших расходов на обслуживание, содержание и ремонт, а доступ к ним был весьма ограничен. В производительности и объеме оперативной памяти основная масса ЭВМ уступала современному смартфону. Все это делало каждый час работы машинного времени очень дорогим, а уровень программистов оценивался их способностью писать те самые «эффективные» программы.

Появление персональных компьютеров изменило ситуацию коренным образом. Вычислительная техника стала надежной и общедоступной, а ее цена, расходы на обслуживание и ремонт – небольшими. Стоимость машинного часа «персоналки» сейчас оказывается ничтожной в сравнении со стоимостью часа работающего с ней человека. И программиста, в том числе. Программирование из элитарной профессии превратилось в обыденное дело и в обязательном порядке изучается каждым школьником. Изменились технологии программирования, появились новые алгоритмические языки.

В этих условиях на первое место выходит уже не критерий эффективности программного продукта, а трудозатраты на его разработку и последующее сопровождение – именно они определяют конечную стоимость программного обеспечения. Фирмы-разработчики с мировым именем бесцеремонно заявляют: «Программа медленно работает? Купите более современный компьютер или попробуйте нарастить оперативную память».

А что же с эффективностью? Она по-прежнему важна при разработке системного программного обеспечения, задач, работающих в реальном времени, программ для микропроцессоров и суперкомпьютеров. Многие ли программисты этим работают? Возможно, 1% или даже 0.1% от общего их количества. Остальные спокойно создают программы, требующие для работы десятков мегабайт памяти, в то время как такие же программы на ЭВМ третьего поколения успешно работали в 64 Кбайтах.

Но все же, – мы для компьютеров или компьютеры для нас? Чье время и силы дороже – человека или машины? Если программа «съест» лишний килобайт памяти, а процессор затратит на вычисление лишнюю десятую долю секунды, но при этом человек высвободит полчаса при создании программы – неужели это так плохо и неправильно?

Требование «эффективности» сейчас – это в первую очередь требование писать программу так, чтобы ее было легко читать, понимать, отлаживать и сопровождать. Чтобы программа, написанная одним человеком, могла быть в короткие сроки понята и модифицирована другим. А разговоры об экономии памяти и времени исполнения – они удел отдельных авторов, написавших школьные (и не

только) учебники еще в эпоху ЭВМ третьего поколения и регулярно переиздающих свои творения без существенных изменений. Вызывает лишь горькую иронию обучение работе в «системе программирования Турбо Паскаль», благополучно скончавшейся четверть века назад.

Мы не будем в дальнейшем заниматься анализом «эффективности по памяти и по времени» - разработка алгоритмов выходит за пределы данной книги, а направим усилия в сторону написания за минимальное время коротких и понятных другим программ. Таковы требования сегодняшнего времени.

Чего вы здесь не найдете

Эта книга не научит вас работать с классическим языком Паскаль, представленным Н.Виртом, и прочими реализациями базового Паскаля, – она с самого начала предполагает изучение языка PascalABC.NET. Вы можете продолжать работать с базовым Паскалем – он достаточно полно поддерживается средой, но его синтаксис и семантика упоминаются лишь там, где это сочтено необходимым.

Вы не научитесь составлять базовые алгоритмы и рисовать блок-схемы – для этой цели существует множество других превосходных книг.

Здесь нет описания архитектуры и схемотехники компьютеров, не рассматриваются основы двоичной арифметики: автор не ставил задачи попытаться заменить школьные учебники информатики.

Как пользоваться этой книгой

Книга рассчитана не только на новичков, начинающих изучать программирование на Паскале, но также и на читателей, обладающих опытом работы с этим языком. Первые научатся писать программы на PasacalABC.NET, вторые получат возможность повысить свой уровень владения языком, освоить современные технологии и приемы программирования.

> Главное условие успешного усвоения материала – занятия перед компьютером с загруженной средой программирования PascalABC.NET версии не ниже 3.4. Настоятельно рекомендуется выполнять все примеры программ, заключенные в рамку. Тексты этих программ, за исключением примитивных, прилагаются.

Предполагается, что пользователь умеет работать с компьютером, способен установить на нем необходимые программные продукты и компоненты, обладает базовыми знаниями информатики на уровне требований учебной программы 6-7 класса средней школы. Для понимания многих задач может понадобиться знание математики.

 Материал предназначен для начинающих.

 Материал рассчитан на читателя, имеющего опыт в программировании.

 Материал требует определенных знаний математики.

В тексте будут встречаться слова и словосочетания, выделенные ***полужирным курсивом***. Это указывает на первое появление нового термина, необходимого для понимания дальнейшего материала. Если термин описан в другом месте, за ним следует ссылка на описание, заключенная в круглые скобки. Отсутствие такой ссылки подразумевает, что термин или должен быть читателю знаком, или разъясняется на месте.

Все приведенные в книге тексты программ, даже если они рассматриваются как варианты для базового Паскаля, работоспособны в PascalABC.NET, если прямо не сказано иное.

Если вы не можете сформулировать на естественном языке алгоритм нахождения минимума из трех чисел и никогда не видели блок-схем – эта книга не для вас!

Парадигмы программирования

PascalABC.NET – современный мультипарадигменный язык программирования. Это означает, что на нем можно писать программы, сочетая различные ***парадигмы***. Не вдаваясь в подробности, будем считать, что под парадигмой понимается некий набор понятий, образующих стиль написания программы.

Ранние универсальные алгоритмические языки базировались на ***императивной*** парадигме, предполагающей запись программы в виде набора инструкций, которые нужно последовательно выполнить для получения результата. Императивная парадигма отвечает на вопрос о том, ***как*** следует решать проблему. Противоположностью является ***декларативная*** парадигма, в которой указывается, что задано и каким должен быть результат. Декларативная парадигма отвечает на вопрос о том, ***что*** нужно сделать для решения проблемы. Возможность сочетать в программе обе эти парадигмы делает программирование комфортным, позволяя не отвлекаться на мелочи.

Одна из первых предложенных парадигм – ***процедурная***. В процессе императивного программирования в последовательных участках кода выделяются некоторые блоки, к которым происходит обращение более одного раза. Эти блоки выделялись в языковые конструкции, получившие название процедур. В процедурной парадигме программа представляет собой набор процедур, одна из которых является главной и из нее производятся обращения к прочим.

Развитие процедурного программирования привело к возникновению парадигмы ***структурного программирования***. В ее основе лежит представление программы, как совокупности процедурных блоков, которая имеет четкую иерархию. Это позволяет лучше видеть всю структуру связей между отдельными блоками, одновременно предполагая, что внесение изменений в один из них не влияет на работу остальных.

Парадигма ***объектно-ориентированного*** программирования предполагает взгляд на программу, как на совокупность отдельных объектов, определенным образом взаимодействующих друг с другом. При этом, каждый объект создается на основе своеобразного «чертежа» - класса.

Все упомянутые парадигмы применяются в императивном программировании. Этим подходам противопоставляется парадигма ***функционального программирования***, трактующая реализацию алгоритма, как процесса нахождения значений некоторых математических функций. Подробнее этот вопрос будет рассмотрен далее.

Возвращаясь к PascalABC.NET можно сказать, что у программиста есть возможность выбрать одну любую парадигму и придерживаться ее, либо в достаточно произвольной пропорции использовать в программе часть или даже все из вышеперечисленных парадигм, получая при этом качественный и наглядный программный код с минимальными затратами труда.

Наша первая программа

В языке Паскаль в простейшем случае программа начинается с английского слова **begin** (начало) и заканчивается словом **end**. (конец) – именно так, с точкой на конце. Конструкция begin … end называется ***операторными скобками***, поэтому вполне допустимо сказать, что простейшая программа – это операторные скобки, за которыми следует точка. Можно набрать такую программу и даже запустить ее. Конечно, она ничего не выведет (и ничего не сделает), но и сообщений об ошибке выдано не будет.

```
begin
end.
```

Формат записи текста программы свободный, и это означает, что его размещение может быть произвольным, например, даже таким: **begin end**. Полезно знать, что у программистов сложился определенный стиль записи текстов программ, который облегчает восприятие программы и ее отладку. Подробнее об этом можно прочитать в Приложении. Выделенные в тексте жирным шрифтом слова программы называются ***ключевыми*** (служебными словами, зарезервированными в языке для нужд программы).

Конструкции языка Паскаль называются ***операторами***. Это нестрогое, но вполне функциональное определение, а разбор подробностей может завести очень далеко.

Операторы разделяются символом «точка с запятой». Перед **end** указывать точку с запятой не требуется, но если вы до этого программировали на языках семейства C (произносится «си»), – можете упорно продолжать ее ставить. Каждый оператор обычно располагают с новой строки, если нет серьезных причин разместить в одной строке несколько операторов.

Если в операторные скобки заключить некоторый набор операторов, получится ***составной оператор*** или ***блок***. Блок в программе может использоваться везде, где может использоваться одиночный оператор языка. Получается, что программой может быть блок, завершающийся точкой.

Наша первая программа будет выводить на монитор фразу «Привет! Я - PascalABC.NET».

```
begin // p01a
  Println('Привет! Я - PascalABC.NET')
end.
```

Слово **begin** и соответствующее ему **end** принято располагать на отдельных строках строго друг под другом, если нет какой-то причины отходить от такого расположения. Это операторные скобки и они, как любые скобки, должны быть парными, т.е. каждой «открывающей скобке» **begin** соответствует своя «закрывающая скобка» **end**, что подчеркивается таким ступенчатым расположением. Все, что находится в операторных скобках, записывается с отступом. Такое расположение удобно делать при помощи клавиши табуляции «Tab».

Оператор Println(‘текст’) организует вывод на монитор текста, записанного в одинарных кавычках, после чего осуществляется переход к следующей строке. Чтобы не делать такого перехода, вместо Println используется Print.

В любом месте программы можно располагать пояснения, которые называются ***комментариями***. Существуют несколько способов записи комментариев:

- комментарий можно начать символами //, например

```
// комментарий – все до конца строки;
```

- комментарий можно начать символом: {, например

```
{ комментарий – все,
        пока не встретится }
```

Фигурные скобки { … } можно заменить на (* … *).

А теперь введите текст программы и запустите ее нажатием клавиши F9.

Если нужно вывести несколько строк, пока что для каждой строки будем записывать отдельный оператор Println.

```
begin // p01b
  Println('Привет! Я - PascalABC.NET'); // первая строка вывода
  Println('-------------------------')  // вторая строка вывода
end.
```

Символы подчеркивания были специально перенесены на другую строчку и выравнены по левой одинарной кавычке. Это позволяет увидеть, как будет оформлен вывод и одновременно избавляет от необходимости подсчитывать количество выводимых символов.

Отметим, что если оператору Print передать не один элемент, а несколько, разделенных запятыми, то после вывода каждого элемента будет делаться пробел.

```
Print(Элемент1, Элемент2, … ЭлементN);
Print('Маша', 'ела', 'кашу');
```

В случае, если пробелы между выводимыми элементами не нужны, вместо оператора Print (Println) используется оператор Write (Writeln). Можно в одном операторе Print (Write) организовать вывод в несколько строк. Для этого в месте, где нужна смена строки, надо записать элемент с именем Newline.

```
Println('Маша', NewLine, 'ела', Newline, 'кашу');
```

Здесь все три слова будут выведены в столбик. Любители стиля «ретро» могут вместо Newline (англ. – новая строка) писать известную в базовом Паскале комбинацию символов #13#10.

Понятие о типах данных

Любые данные в программе на языке Паскаль перед своим первым использованием обязательно должны быть объявлены. Объявление состоит в присваивании данным имен и указании их особой характеристики – ***типа данных***.

Тип данных указывает на то, как данные должны быть организованы для хранения в памяти компьютера, а также определяет правила их обработки. Паскаль относится к языкам со строгой типизацией. Это означает, что тип данным присваивается обязательно и не может быть впоследствии изменен.

Приведенная классификация типов данных в PascalABC.NET не является исчерпывающей. Отдельные типы данных могут компоноваться в группы подобно тому, как это указано для порядковых типов.

Тип данных называется ***структурированным***, если в одной переменной этого типа может содержаться множество значений. Таковы массивы, множества, строки и т.д.

Особым типом данных является ***последовательность***, которая по существу хранит алгоритм последовательного получения данных.

.

Все типы, кроме типов указателей, являются производными от типа **Object**. Каждый тип в PascalABC.NET имеет отображение на тип .NET Framework. Тип указателя принадлежит к неуправляемому коду и моделируется типом **void***.

Следует понимать, что приведенная классификация является достаточно условной. Несмотря на то, что PascalABC.NET по возможности сохраняет поведение типов, которые пришли из языка Object Pascal (Delphi), имеется ряд особенностей реализации, связанных как с платформой .NET, так и с самим PascalABC.NET. Об этих особенностях будет подробно сообщаться в соответствующих главах книги.