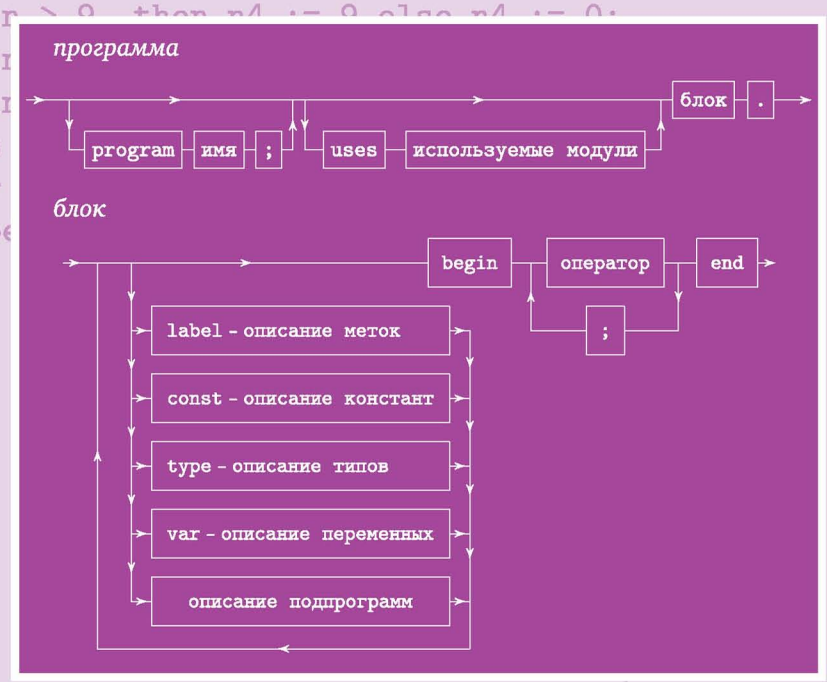
Е. В. АНДРЕЕВА

ПРОГРАММИРОВАНИЕ – ЭТО ТАК ПРОСТО



ПРОГРАММИРОВАНИЕ – ЭТО ТАК СЛОЖНО

СОВРЕМЕННЫЙ УЧЕБНИК ПРОГРАММИРОВАНИЯ

Е. В. Андреева

Программирование — это так просто, программирование — это так сложно

Современный учебник программирования

Издание второе, исправленное и дополненное

Москва
Издательство МЦНМО
2015

УДК 519.671
ББК 22.18
А65

Андреева Е. В.

А65 Программирование — это так просто, программирование — это так сложно. Современный учебник программирования. — М.: МЦНМО, 2015. — 2-е изд., испр. и доп. — 184 с.

ISBN 978-5-4439-0259-3

Книга представляет собой практический курс для обучения программированию, основную часть которого составляет подборка около 200 задач. В ней делается попытка показать, как обучить программированию в школе за 16 уроков. Рассмотрены все алгоритмы из перечня, входящего в «Требования к уровню подготовки выпускников» согласно нормативным документам ЕГЭ.

Большинство приведенных задач предполагают проверку их решений на системе тестов. Автоматическая проверка решений задач будет организована на сайте `informatics.mccme.ru`.

Книга предназначена для учителей информатики и старшеклассников, изучающих информатику на профильном уровне или готовящихся к ЕГЭ по информатике. Пособие может быть использовано на кружковых и факультативных занятиях, а также в школах с углубленным изучением математики и информатики при изучении программирования учениками 8—9 классов.

Первое издание книги вышло в 2009 г.

ББК 22.18

Андреева Елена Владимировна

Программирование — это так просто,
программирование — это так сложно

Подписано в печать 10.11.2014 г. Формат $60 \times 90\ 1/16$. Бумага офсетная.
Печать офсетная. Печ. л. 11,5. Тираж 2000 экз. Заказ № .

Издательство Московского центра
непрерывного математического образования.
119002, Москва, Большой Власьевский пер., д. 11.

Отпечатано с готовых диапозитивов в ППП «Типография „Наука“».
121099, Москва, Шубинский пер., 6.

Книги издательства МЦНМО можно приобрести в магазине «Математическая книга»,
Большой Власьевский пер., д. 11. Тел. (499) 241–72–85. E-mail: `biblio@mccme.ru`

ISBN 978-5-4439-0259-3

Оглавление

Предисловие

Введение курса информатики в школах нашей страны фактически начиналось с преподавания программирования. В то время даже был провозглашен лозунг: «Программирование — это вторая грамотность». Заметим, что компьютеры в школах тогда практически отсутствовали. А та техника, которой оснащались школы в конце 80-х — начале 90-х годов прошлого века, практическую составляющую курса информатики все равно невольно сводила к программированию.

Одновременно с революционным развитием аппаратного и программного обеспечения и оснащением современной компьютерной техникой учебных заведений курс информатики претерпел существенные изменения. Основное внимание в большинстве школ стало уделяться освоению современных информационных технологий. Эти тенденции отражены и в новом Стандарте по информатике, в котором собственно обучению программированию отводится очень мало времени. Но, как заметил автор школьных учебников по информатике А. Г. Гейн, «...очевидно, что именно алгоритмизация с самого начала вытянула на школьную арену курс информатики и ныне во многих реально существующих курсах информатики позволяет уйти от умных, но пустоватых разговоров к конкретному делу (не случайно альтернативой алгоритмизации нередко выступает обучение информационным технологиям — учить детей тому и другому многим представляется невозможным, ибо освоение реального дела требует значительных затрат и труда, и времени)».

Кого следует учить программированию

В рамках часов, отводимых Примерной программой в базовом курсе информатики на алгоритмизацию и программирование, овладение даже основами программирования на современных алгоритмических языках представляется невозможным. Тем не менее, контингент школьников, у которых интерес именно к изучению, а не знакомству с программированием высок, несомненно, существует. В первую очередь это учащиеся физико-математических школ, гимназий, лицеев и гимназических классов общеобразовательных школ. У большинства из них есть как мотивация, так и способности к освоению программирования. Учебные планы подобных образовательных учреждений, в которых на освоение информатики и информационных технологий отводится не менее часа начиная с 5-го класса, не

менее двух — с 8-го и до четырех часов в 10—11 классах, также играют положительную роль. Мотивация есть и у учителя — ведь большинство современных олимпиад по информатике являются по своей сути олимпиадами по программированию, а по успехам учеников в олимпиадах зачастую судят о квалификации учителя, хотя в случае с информатикой это далеко не бесспорно. Кроме того, любовь к программированию многие учителя информатики принесли из своей профессиональной деятельности, и, конечно же, им хочется передать эту любовь и своим ученикам.

Таким образом, очевидно, что существуют школы, в которых могут и хотят учить школьников программированию, и естественно возникают вопросы — в каком возрасте начинать обучение и какой язык программирования лучше использовать? Ответ на первый вопрос не очень сложный. Если в 5—6 классах школьники уверенно освоили большинство современных информационных технологий: текстовый процессор (редактор), программу для создания презентаций, иногда — электронную таблицу (здесь намеренно не упоминается графический редактор, так как его освоение в простейшей форме сейчас происходит в основном уже в начальной школе, а профессиональные редакторы типа PhotoShop заслуживают детального рассмотрения либо в более старших классах, либо на факультативах), то уже в 7—8 классах можно попытаться построить курс информатики так, что его стержнем окажется именно изучение программирования. При этом теоретическая часть программы базового курса информатики не только не будет проигнорирована, но и, наоборот, будет освоена учащимися на более глубоком, практическом уровне.

В основе предлагаемого курса программирования лежит многолетний опыт работы автора в физико-математической школе-интернате им. А. Н. Колмогорова (СУНЦ МГУ) и с учениками 8—9 классов школы «Интеллектуал» г. Москвы. Изучение языка Logo или аналогичного ему в младших классах средней школы, конечно же, является чрезвычайно полезной пропедевтикой данного курса, однако непосредственно опираться на него мы не будем. Начинать преподавание программирования можно как в 8-м (иногда 7-м), так и в 9—10 классах, при этом незначительно меняются решаемые на уроках задачи, которые должны быть адаптированы к уровню математической подготовки учащихся. Отметим, что полноценные занятия можно проводить лишь тогда, когда на уроки информатики отводится не менее двух (спаренных) учебных часов в неделю. В противном случае изучение программирования лучше проводить в рамках факультатива.

Выбираем язык программирования

Перейдем к обсуждению выбора языка программирования. Здесь, на наш взгляд, категоричной рекомендации быть не может. Единственное условие, которое должно выполняться, — это то, что не только язык, но и выбранная среда программирования должны быть одними из реально используемых в современной практике, в том числе и профессионалами. Обратимся к мировому опыту как обучения программированию, так и профессионального программирования.

Basic, Quick Basic и Visual Basic

Долгие годы считалось, что язык программирования Basic является с методической точки зрения непригодным для обучения даже началам программирования будущих профессионалов, так как программирование с GoTo приводит к формированию плохого стиля, исправить который в дальнейшем очень сложно. Однако эволюция языка, начиная с Quick Basic и заканчивая Visual Basic, привела к тому, что сам язык стал мало отличаться, например, от языков Pascal и Delphi соответственно, и категоричные высказывания о непригодности его использования в учебных целях вряд ли можно считать корректными. Тем не менее, при выборе этого языка следует учитывать, что на многих олимпиадах высокого уровня по информатике и программированию в списке допустимых языков программирования Basic либо отсутствует вовсе (международная олимпиада школьников по информатике, студенческий чемпионат мира по программированию), либо присутствует в качестве одной из версий, зачастую весьма отличной от той, что изучалась в школе (так, в школе часто изучают Quick Basic, а на всероссийской олимпиаде в последние годы разрешенным является уже Visual Basic, причем написание корректных консольных приложений на данном языке даже у знакомых с ним школьников вызывает большие трудности).

Если же обратиться к статистике использования тех или иных языков программирования на международном рынке труда, то Visual Basic в настоящее время занимает почетное второе место, уступая лишь Visual C++. Однако это далеко не так у нас в стране, где аналогичную Visual Basic программистскую нишу прочно занимает Delphi. Единственным неоспоримым аргументом в пользу выбора Visual Basic в качестве базового для изучения программирования вообще является то, что именно этот язык используется для написания макрокоманд в современных офисных приложениях. И если изучение, например, электронных таблиц ведется на уровне, предполагающем свободное владение этим языком, то выбор Visual Basic для освоения алгорит-

мизации и программирования может быть оправданным. В данном случае оказывается возможным построить интегрированный курс одновременного освоения как информационных технологий, так и программирования. Многие учителя отмечают также легкость «быстрого старта» при знакомстве школьников с данным языком. Но мой опыт работы показывает, что школьники, способные к овладению программированием вообще, очень быстро преодолевают сложности работы с любой средой программирования и легко осваивают формальные правила записи программ на изучаемом языке, в частности описание переменных с корректным указанием их типов.

Итак, большинство аргументов «за» и «против» языка Basic рассмотрены. Перейдем к анализу языков группы Pascal и сравнению их с «Си-подобными» языками.

Pascal или C (C++)

Напомним, что язык Pascal был создан в начале 70-х годов прошлого столетия выдающимся специалистом в области computer science Никлаусом Виртом именно как язык для изучения программирования. Основой для построения синтаксических конструкций этого языка послужил широко распространенный в то время Algol (**algo**rithmical **l**anguage). Вирт продолжил свою работу над созданием методически обоснованного языка программирования, предложив общественности сначала язык modula-2, а затем объектно-ориентированный Oberon. Однако последние два языка не получили сколь-либо широкого распространения в отличие от языка Pascal, чрезвычайной популярности которого способствовало развитие семейства компиляторов фирмы Borland, начиная от Turbo Pascal и заканчивая Delphi. Не все новшества, привнесенные специалистами Borland в классический Pascal, кажутся Вирту оправданными, тем не менее, и они, в том числе, привели к тому, что Pascal долгие годы занимал одно из ведущих мест среди профессиональных языков разработки различных приложений, а проект Delphi придал ему новое дыхание.

С методической точки зрения Pascal действительно хорошо подходит на роль учебного языка. Он позволяет познакомиться с большинством понятий современного программирования, освоить как различные типы, так и структуры данных. Программы на Pascal легко читаются, а один из важнейших принципов современного программирования — «удобочитаемость более важна, чем краткость кода» (конечно, если это не приводит к замедлению работы программы более чем в два-три раза), ведь над современными программными комплексами трудятся целые коллективы программистов, и им

необходимо быстро ориентироваться в коде друг друга. Не случайно при описании различных алгоритмов в большинстве учебной литературы, в том числе и западной, используется именно Pascal или схожий с ним псевдокод. Удобочитаемость Pascal весьма кстати и учителю при проверке программ, написанных школьниками. Кроме того, синтаксис языка устроен так, что своей строгостью фактически вынуждает писать правильные программы. Это выгодно отличает Pascal, например, от языка C (C++), который, давая программисту широкие возможности, требует от него знания многих нюансов, зачастую упускаемых из вида начинающими программистами. Чего стоит, например, конструкция

```
if (a = b),
```

которая, являясь синтаксически корректной, имеет в C весьма далекую от интуитивного смысла семантику[1] (переменной `a` присваивается значение переменной `b`, затем значение `a` сравнивается с нулем). Или расссмотрим логическое выражение вида

```
a < b < c.
```

Оно является некорректным для числовых и символьных типов языка Pascal и не будет пропущено компилятором, но вполне воспринимается компиляторами с языка C (C++), однако имеет смысл, весьма отличный от двойного неравенства в математике: сначала выполняется сравнение `a` и `b`, результатом которого оказывается либо 1 (истина), либо 0 (ложь), а затем уже это число (0 или 1) сравнивается с `c`. Ситуацию, в которой такое сравнение имеет смысл, придумать практически невозможно. Завершить обзор подобных нюансов (сами же нюансы на этом вовсе не заканчиваются) хочется семантикой оператора `switch` в языке C (C++), которая также зачастую приводит к написанию некорректных программ начинающими программистами, так как только оператор `break` в конце описания каждого из вариантов переключателя позволяет придать данной конструкции тот смысл, в котором она в большинстве случаев используется (только в этом случае она превращается в аналог оператора `case` из других языков программирования).

Не случайно на факультете вычислительной математики и кибернетики МГУ им. М. В. Ломоносова в большинстве групп до последнего времени курс программирования начинался именно с изучения языка Pascal, причем в его классической версии... Казалось бы, на этом

[1] Синтаксис языка определяет правила записи на нем программ, а семантика — что означает та или иная языковая конструкция (например, как должен выполняться тот или иной оператор).

вопрос о выборе языка программирования для современной школы можно считать решенным. Однако не все так просто.

Если обратиться к опыту стран юго-восточной Азии, а именно в них широко ведется преподавание школьникам программирования, то там уже долгие годы обучение старшеклассников проводится на базе языка C++. А, например, в Южной Корее мне удалось наблюдать, как ученики 6—7 классов (вернее, классов, которые соответствуют нашим 6—7, так как корейские школы имеют три ступени, в каждой из которых классы нумеруются от 1) уверенно программировали на Java. Правда, происходило это в воскресной школе дополнительного образования при одном из местных университетов, но подобные формы обучения вполне соответствуют нашим кружкам или факультативам. Кроме того, насколько мне известно, еще в 2001 г. болгарские специалисты в области преподавания информатики в средних и высших учебных заведениях подготовили новые учебники, предназначенные для массового перевода школ страны на преподавание языка C++ вместо Pascal. В нашей же стране примерно в это же время осуществлялся массовый отход от преподавания программирования вообще. Выбор профессионалов в последние годы также лежит между C++ и Java. Более того, все так называемые скриптовые языки и языки web-программирования имеют C-подобный синтаксис. А для операционных систем UNIX-класса именно язык C является фактически родным. Наступление на Pascal ведется и на олимпиадном фронте: его использованию на студенческих чемпионатах мира, видимо, пришел конец (а разрешенными останутся как раз только C++ и Java). Все это вновь заставляет задуматься о выборе языка программирования.

В нашей стране также есть школы, в которых весьма успешно ведется преподавание программирования на языке C (или C++). В основном это физико-математические школы (например, школа № 30 г. Санкт-Петербурга). Большую роль здесь играет как контингент учащихся, так и личность учителя. В ряде школ C изучают как второй после Pascal язык (например, в гимназии № 1514 г. Москвы).

Мы также пытались вести преподавание программирования на языке C++, по крайней мере для тех из наших учеников, которые к 10 классу уверенно программировали на языке Pascal. К сожалению, этот опыт оказался не слишком удачным.

Помимо уже упомянутых выше сложностей, возникающих при написании программ на C++, особенно на начальном этапе его освоения, сообщения компилятора о различного рода ошибках в программе на C++ не столь информативны, как в средах программирования для языка Pascal фирмы Borland и ее приемников.

Pascal или Python

В первом издании язык программирования Python не обсуждался. За прошедшее время автор приобрел опыт обучения школьников 7—9 классов программированию именно на этом языке. И этот опыт оказался крайне удачным, даже по сравнению с Pascal. Порог вхождения в него сопоставим с Quick Basic, а возможности и распространенность в профессиональной среде существенно выше.

Главным недостатком выбора языка Python в качестве первого учебного языка программирования коллеги называют динамическую типизацию и сокрытие информации о представлении данных в компьютере. Так, для целых чисел в Python используется и стандартная компьютерная арифметика, и так называемая «длинная», при этом программисту не надо беспокоиться о выборе одной из них, как в JAVA, — интерпретатор сам примет правильное решение. Но этот так называемый недостаток для большинства программистов на Python, наоборот, кажется достоинством. Реальный же недостаток, который не позволяет автору полностью вести обучение программированию только на Python, — это быстродействие существующих интерпретаторов. Из-за невысокой скорости выполнения программ на Python решение, например, олимпиадных задач по информатике начиная с какого-то уровня, становится невозможным. Хотя призером регионального этапа всероссийской олимпиады стать можно, а муниципальный этап писать на языке Python заведомо проще и результативней.

Тем не менее Python хорошо подходит в качестве единственного языка программирования тем школам, где обучение программированию ведется ознакомительно или в рамках подготовки к ЕГЭ по информатике. И следующие материалы автора по обучению программированию, несомненно, будут уже на этом языке. В настоящее время на сайте `informatics.msk.ru` выложен авторский курс учителя школы № 179 г. Москвы Д. П. Кириенко, включающий в себя начальное описание языка Python 3 и множество задач как для начинающих, так и для сильных учащихся.

Версии Pascal

В настоящее время в российских школах используется несколько различных версий языка Pascal. Долгое время наиболее распространенной была среда программирования Borland Pascal 7.0 (BP 7.0). BP 7.0 был выпущен в 1993 г., то есть этой замечательной среде программирования уже больше 20 лет. Фирмой Borland она давно

не поддерживается, в результате чего она все меньше совместима с современными высокоскоростными компьютерами и операционными системами. Ограничения, налагаемые языком Borland Pascal на размер используемой памяти, возникшие в результате ориентации на DOS-модель памяти, не выдерживают никакой критики (64 Кб на все глобальные переменные, столько же на размер стека и 200—400 Кб на динамические переменные, редко используемые в школьных курсах). Ну и, наконец, невозможность простого и быстрого создания оконных приложений с различными элементами интерфейса, отвечающих современным требованиям.

Все эти проблемы были решены при создании среды Delphi (для работы в школе можно рекомендовать любую версию начиная с 6-й), однако коммерческий характер этой среды постепенно делает нереальным ее использование в российских школах. На международном уровне, в том числе на международной олимпиаде по информатике, де факто стандартом стал компилятор Free Pascal и соответствующие ему среды. Так, бесплатным аналогом среды Delphi является Lazarus. И практически все относящееся в данном издании к Delphi применимо также и Free Pascal. Ознакомиться со средой Lazarus можно на официальном сайте `www.lazarus.freepascal.org`.

Спешим успокоить тех, кому пока морально или технически тяжело перейти на подобный объектно-ориентированный язык с визуальной средой программирования: наш курс не будет опираться на особенности такой среды в частности и визуального программирования вообще. Рассматриваться будут лишь задачи, решаемые в рамках так называемых *консольных* (в отличие от *оконных*) приложений, требующие предварительного нахождения *алгоритма* решения. Особенности языка Object Pascal (то есть языка Delphi) будут лишь изредка упоминаться в сравнении с аналогичными в языке Borland Pascal.

Другой альтернативой для многих стала российская версия среды и компилятора с языка Pascal — PascalABC. В нем подкупает русскоязычный интерфейс, удобство работы в операционной системе Windows, простота в создании графических приложений и многое другое. Тем не менее данная среда лишь недавно стала предлагаться участникам официальных олимпиад по программированию, компилятор с этого языка не поддерживает другие версии Pascal, а разработчики языка пошли по пути его совершенствования, что делает программы, написанные для PascalABC, непереносимыми для других компиляторов. Наконец, данная среда совсем не известна в мире, что в каком-то смысле делает ее вещью в себе. Поэтому мы не будем в данном издании разбирать возможности этой версии языка Pascal.

Два подхода к изучению программирования

Пусть выбор языка и среды программирования сделан. Рассмотрим теперь два подхода к изучению языка программирования: *формальный* и «программирование по образцу». Первый основан на формальном (строгом) описании конструкций языка программирования (*синтаксиса* языка и его *семантики*) тем или иным способом (с помощью синтаксических диаграмм, мета-языка или формального словесного описания) и использовании при решении задач только изученных, а следовательно, понятных элементов языка. При втором подходе школьникам сначала выдаются готовые программы, рассказывается, что именно они делают, и предлагается написать похожую программу или изменить имеющуюся, не объясняя до конца ряд «технических» или несущественных с точки зрения учителя для решения задачи деталей. При этом говорится: «Точный смысл соответствующих конструкций вы узнаете позднее, а пока поступайте аналогичным образом». Второй подход дает возможность так называемого «быстрого старта», но создает опасность получить полуграмотных пользователей среды программирования, то есть людей, которые используют в своей практике достаточно сложные конструкции, но не могут четко объяснить, почему в том или ином случае нужно применять именно их и как они работают. В результате рано или поздно такие «программисты» сталкиваются с ошибками, исправить которые они просто не в состоянии, — им не хватает знаний.

В своей практике при работе с десятиклассниками мы используем в основном первый, формальный подход. При этом некоторыми неформальными умениями эти школьники чаще всего уже обладают. На наш взгляд, одна из задач школьной информатики — научить именно формальному подходу, в частности при применении различных определений. И формальное изучение языка программирования этому немало способствует. Но и без хороших примеров (образцов) при обучении программированию школьников не обойтись. И чем младше ученики, тем больше примеров необходимо приводить при описании языка (иногда даже заменяя ими строгое определение). Другое дело, что, на наш взгляд, следует добиваться того, чтобы в результате обсуждения примера все его детали оказались понятны школьникам (обязательно нужно объяснить, как и почему это работает, в том числе опираясь на уже изученный формальный материал). В этом случае сильные ученики получат возможность понять все досконально и смогут использовать полученные знания в дальнейшем, а средние — приобретут конкретные навыки.

Переменные

Любой алгоритм или программа для компьютера состоит из двух разделов: описания *данных* и описания *действий*, которые над этими данными необходимо выполнить. В языках программирования действия представляются так называемыми *операторами*. Данные есть общее понятие для всего того, с чем оперирует компьютер. Память компьютера с точки зрения языка Pascal разделена на секции, называемые *переменными*. Переменные бывают разных *типов*. Тип определяет размер памяти, отводимой под переменную, а также допустимое конечное множество значений, которые может принимать та или иная переменная. Каждая переменная имеет жестко закрепленное за ней имя. Значения переменных могут меняться в ходе выполнения программы.

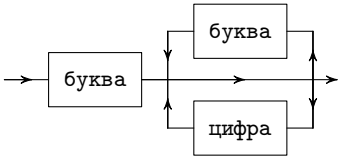
Для обозначения имен переменных, констант, типов, процедур, функций и самой программы используются *идентификаторы*.

Идентификатор — это любое количество *букв* и *цифр*, начинающееся с *буквы*. Идентификатор не может совпадать со служебным словом. Имена могут быть очень длинными, но в конкретных реализациях в них анализируется лишь несколько первых символов (например, 63, то есть имена, у которых первые 63 символа совпадают, считаются одинаковыми). Прописные и строчные буквы в именах не различимы. Так, имена `my_name`, `My_Name` и `mY_nAmE` совпадают.

Список служебных (или зарезервированных, или ключевых) слов (на примере Delphi):

`and`	`exept`	`label`	`set`
`array`	`exports`	`library`	`shl`
`as`	`file`	`mod`	`shr`
`asm`	`finalization`	`nil`	`string`
`begin`	`finally`	`not`	`then`
`case`	`for`	`object`	`threadvar`
`class`	`function`	`of`	`to`
`const`	`goto`	`or`	`try`
`constructor`	`if`	`packed`	`type`
`destructor`	`implementation`	`procedure`	`unit`
`dispinterface`	`in`	`program`	`until`
`div`	`inherited`	`property`	`uses`
`do`	`initialization`	`raise`	`var`
`downto`	`inline`	`record`	`while`
`else`	`interface`	`repeat`	`with`
`end`	`is`	`resourcestring`	`xor`

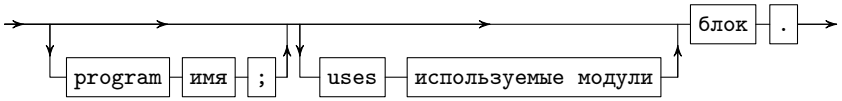
Для описания термина *идентификатор* очень удобно использовать синтаксическую диаграмму:



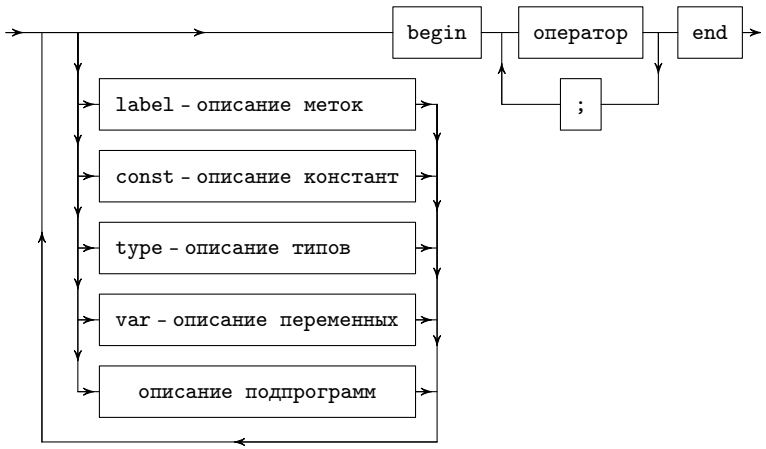
Синтаксическая диаграмма — это графическое описание синтаксиса языка программирования. С ее помощью можно определить корректность тех или иных конструкций (предложений) с точки зрения конкретного языка программирования. В отличие от блок-схем, у синтаксической диаграммы могут существовать безусловные разветвления, что означает при проверке синтаксиса возможность пойти по любой из ветвей. Могут присутствовать и бесконечные циклы. Так, из приведенной диаграммы следует, что букв и цифр в одном имени может быть как угодно много, а после первой буквы может находиться буква, цифра или ничего больше не стоять (конец диаграммы). В дальнейшем буквы и цифры могут также чередоваться произвольным образом.

Общий вид программы

программа



блок



Урок 1

Простейшая программа на языке Pascal

Собственно программа на языке Pascal (как еще говорят, *тело программы*) должна начинаться со слова `begin`, а заканчиваться словом `end` и знаком точки (см. синтаксическую диаграмму), то есть ее можно сравнить с законченным предложением. Программа, состоящая только из этих слов, разделенных пробелом или переводом строки, верна, но ничего не делает. Добавим в нее вызов процедуры печати каких-либо сообщений (в дальнейшем мы будем называть его *оператором вывода*), например:

```
begin
  write('Hello!')
end.
```

Текст, заключенный в апострофы, компьютер не анализирует, а просто выводит на экран, поэтому он может быть произвольным, то есть содержать любые символы, набираемые на клавиатуре, в том числе русские и латинские большие и маленькие буквы. Программа может быть записана и в одну строку, тогда различные слова нужно разделять хотя бы одним пробелом:

```
begin  write('Hello!') end.
```

Наберите текст этой программы и запустите ее на выполнение. Научитесь просматривать результат работы программы. Иногда для этого удобно вставлять в конец программы оператор `readln`.

Теперь познакомимся с правилами вывода данных в языке Pascal подробнее.

Вывод данных

Стандартные процедуры `write` и `writeln` служат для вывода информации. Правило их использования одно и то же: после слова `write` или `writeln` в скобках через запятую перечисляются параметры, которые мы хотим напечатать. Число этих параметров не ограничено. Запятая служит разделителем между параметрами:

```
writeln(параметр, параметр, ..., параметр)
```

Существует три вида параметров: *константы*, *переменные* и *выражения* (например, арифметические выражения). Константы быва-

ют числовые (это просто различные числа — целые и вещественные), логические и строковые. Любой текст, набранный с клавиатуры и заключенный в апострофы (одиночные кавычки), называется *строковой константой*. Если в текст нам нужно поместить апостроф, например в слове O'key, на этом месте нужно набить два апострофа подряд вместо одного: `write('O''key')`. Все параметры в `write` или `writeln` независимы друг от друга, поэтому в одном и том же операторе могут встречаться параметры разных типов в произвольном порядке.

При выполнении оператора вывода все параметры будут напечатаны в одной строке в том же порядке, в каком они перечислены в списке параметров. Любая константа, числовая или строковая, будет напечатана так, как вы ее написали в вызове `write` или `writeln` (в строковой константе начальный и конечный апострофы напечатаны не будут, а вместо двух апострофов, расположенных в строковой константе подряд, на экране появится в этом месте один); вместо переменной на экране появится ее значение, а вместо арифметического выражения — результат его вычисления.

Между `write` или `writeln` существует единственное различие: после (!!!) выполнения `writeln` курсор переходит на новую строку, а после выполнения `write` курсор остается в той же строке, и новая печать данных с помощью `write` или `writeln` или ввод данных при помощи `read` или `readln` (операторов чтения данных) будет проходить в той же строке.

При печати параметров пробелы между ними автоматически не вставляются, например, при печати чисел 1, 2, 3 с помощью `writeln(1, 2, 3)` все они сольются в одно число — 123. Чтобы разделить выводимые элементы, можно поместить между ними символ пробел, например `writeln(1, ' ', 2, ' ', 3)`, или отформатировать печать, поставив после каждого элемента списка вывода двоеточие и целое число, которое указывает, сколько позиций на экране должна занимать выводимая величина, например `writeln(1:3, 2:3, 3:3)`. Отметим, что элемент дополняется начальными пробелами слева, с тем чтобы соответствовать указанной после двоеточия величине. Результаты выполнения двух последних операторов будут выглядеть так:

```
1 2 3
  1   2   3
```

Если указанное в формате печати число меньше, чем необходимо, то Pascal при выводе увеличит это значение до минимально необходимого размера. При выдаче на экран значений вещественных выра-

жений в формате печати полезно указывать еще один параметр после второго двоеточия. Он будет обозначать количество символов после десятичной точки, которые мы хотим напечатать. Например, при печати результата стандартной функции `pi`, которая с машинной точностью выдает значение числа π, оператор `write(pi:0:0, pi:5:2, pi/2:2:0)` выдаст на экран:

```
3 3.14 2
```

Заметим, что при печати фиксированного количества цифр вещественного числа оно предварительно округляется по правилам математики.

Примеры операторов вывода:

```
write('Нажмите любую клавишу');
writeln(2, '+', 2, '=', 4);
write('7+5', '='); writeln(7 + 5);
```

Задания

1. Определите, что будет выведено на экран после выполнения трех операторов, приведенных в конце урока 1. Напишите соответствующую программу и сверьте результат ее выполнения со своим предположением.

2. Напишите программу, которая семью различными способами будет выдавать на экран фразу «2+2=4» (без кавычек). Воспользуйтесь для этого операторами `writeln`, которые следует разделять друг от друга знаком точка с запятой (;) — разделителем между операторами в языке Pascal.

Первый способ должен содержать всего один параметр в операторе `writeln`, второй — два и т. д., а пятый, шестой и седьмой — пять. При этом шестой оператор не должен содержать числа 4, а седьмой — числа 2. В операторах печати должны использоваться все три вида параметров.

3. Напишите программу, которая выводит на весь экран ваше имя. Линии букв можно составлять из символов «*» или других символов.