Г. Г. Рапаков
С. Ю. Ржеуцкая

Программирование на языке Pascal

- *Базовые понятия программирования*
- *Конструкции языка*
- *Структурное и модульное программирование*
- *Массивы, множества, записи, файлы*
- *Динамические структуры данных*
- *Введение в объектно-ориентированное программирование*

УЧЕБНОЕ ПОСОБИЕ

bhv

УДК 681.3.068+800.92Pascal
ББК 32.973.26-018.1я7
Р23

Рапаков Г. Г., Ржеуцкая С. Ю.

Р23 Программирование на языке Pascal. — СПб.: БХВ-Петербург, 2004. — 480 с.: ил.

ISBN 5-94157-401-0

Учебное пособие ориентировано на широкий круг читателей, как начинающих знакомство с программированием, так и имеющих в нем достаточный опыт. Необходимое для новичков изложение азов предмета сочетается в книге с подробным и глубоким описанием тонкостей языка Pascal. Издание насыщено примерами и содержит множество полезных рекомендаций. Особое внимание уделено вопросам стиля в программировании, как линейном, так и объектно-ориентированном. Каждую главу завершают контрольные вопросы и задания, сложность которых дозированно возрастает от начальных глав к конечным. Эти материалы могут быть рекомендованы преподавателям информатики средней и высшей школ в качестве методических.

Для школьников старших классов и студентов

УДК 681.3.068+800.92Pascal
ББК 32.973.26-018.1я7

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Галина Смирнова*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 20.10.03.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 38,7.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

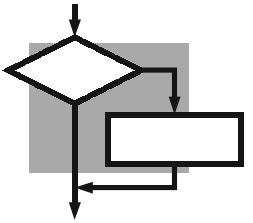
Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-401-0

Содержание

Глава 2



Язык программирования. Программа

2.1. Язык программирования

Алгоритм, представленный в виде блок-схемы или в словесной форме (даже с использованием псевдоязыка), не пригоден для исполнения *процессором* какого-либо вычислительного устройства (компьютера, робота и др.). В этом случае нужен абсолютно формальный способ записи, не допускающий даже малейшей неточности. Следовательно, нужен особый язык, который позволяет записывать алгоритмы с использованием ограниченного набора конструкций, не допускающих неоднозначного толкования. Такие языки называются языками программирования.

Таким образом, *язык программирования* — совокупность средств и правил представления алгоритма в виде, пригодном для выполнения вычислительной машиной.

Рассмотрим кратко историю языков программирования.

2.1.1. Исторический обзор

Машинный язык

На раннем этапе развития вычислительной техники программы писались на машинном языке — в *машинных кодах*, т. е. так, как их воспринимает процессор компьютера или другого цифрового устройства. Для каждой элементарной операции в этом случае требуется указать ее код и адреса ячеек памяти с данными. Запись выполняется в цифровом виде с использованием двоичной системы счисления. Основными неудобствами такого способа программирования являются следующие:

- исторически сложилось так, что имеется много типов процессоров, отличающихся друг от друга *архитектурой* (устройством) и *системой команд* (набором допустимых инструкций). В результате программа на *машинном языке* годится только для исполнения тем процессором, для которого она написана;

- программу на машинном языке трудно читать даже профессионалу. В такой программе тяжело находить ошибки. Если объем программы превышает критический, программу практически невозможно полностью отладить. Даже если программа доведена до уровня, при котором она отвечает поставленной задаче, малейшие изменения могут вызвать непреодолимые трудности.

В настоящее время машинный язык используется только в исключительных случаях для программирования специальных задач с использованием специализированных вычислительных устройств.

Ассемблер

Первоначальный прогресс в технологии программирования был связан с идеей использования символьных имен (названий) вместо цифровых кодов операций и адресов данных. Язык записи команд, основанный на этой идее, получил название языка Ассемблера. Использование осмысленных названий вместо кодов операций и адресов памяти существенно упрощает процесс программирования и внесения изменений в программу. К сожалению, основной недостаток машинного языка — зависимость программы от конкретного типа процессора — при переходе к Ассемблеру не был устранен, но *это был только первый шаг в правильном направлении.*

Программа, записанная на Ассемблере, не может восприниматься процессором непосредственно (ему нужны только двоичные коды операций и адреса). Следовательно, необходимо предварительное преобразование (перевод) программы с языка Ассемблера на машинный язык. Это делается с помощью специальной программы, называемой *транслятором*, а *процесс преобразования программы в машинные коды называется трансляцией (см. разд. 3.1).* Попутно на транслятор были возложены функции выявления некоторых ошибок в тексте программы, нарушающих правила записи. Такие ошибки называются *синтаксическими.*

Языки программирования высокого уровня

Следующая ступень развития — это *языки программирования высокого уровня.* Их использование позволяет отвлечься от системы команд конкретного типа процессора. Такой язык содержит правила записи программ, которые, с одной стороны, достаточны и удобны для описания алгоритмов решения задач, а с другой стороны, толкуются однозначно и могут быть преобразованы в программы в машинных кодах.

Задача программиста заключается в том, чтобы подготовить правильный текст на языке программирования, а остальное возьмет на себя транслятор, который прочтет этот текст, проверит его на соответствие правилам языка и сформирует программу на машинном языке. Естественно, транслятор должен "знать" систему команд своего процессора и особенности работы уста-

новленного на компьютер программного обеспечения, его *операционной системы*. Это приводит к необходимости разработки нескольких различных трансляторов для одного и того же языка программирования.

Также понятно, что разработка транслятора с языка высокого уровня — задача более сложная и дорогостоящая, чем разработка транслятора с Ассемблера, а программа в машинных кодах, сформированная самым лучшим транслятором, все равно будет уступать по качеству и быстродействию программе, разработанной программистом-профессионалом на Ассемблере. За удобство программирования приходится платить некоторым снижением эффективности разработанной программы.

Говорят, что языки программирования высокого уровня являются *машинно-независимыми*. В противоположность этому, машинный язык и язык Ассемблера — *машинно-зависимые языки*. Иногда их называют *языками программирования низкого уровня*. Конечно, слова "низкий уровень" вовсе не означают низкое качество этих языков. Так подчеркивается их неразрывная связь с аппаратурой компьютера. Во многих задачах, например, при разработке ядра (основной части) операционных систем, где качество машинного кода ставится выше удобства разработки, без языка Ассемблера вообще не обойтись. Тем не менее, везде, где возможен компромисс, язык высокого уровня предпочитают Ассемблеру.

Возможность создания независимых от системы команд компьютера языков программирования привела к их бурному развитию. В настоящее время насчитывается более сотни самых разнообразных языков программирования высокого уровня. Большинство из них предназначены для применения в конкретных областях (например, языки программирования баз данных, языки программирования для Internet и т. д.). Эта группа языков развивается очень быстро, т. к. сфера применения компьютеров постоянно расширяется и возникает потребность как в новых языках программирования, так и в развитии уже имеющихся.

Вместе с тем имеется довольно устойчивая группа хорошо отработанных, испытанных языков универсального назначения, очень близких по принципам, заложенным в их основу, и по используемым конструкциям. На протяжении десятилетий эти языки постепенно совершенствовались, а заодно сближались, перенимая друг у друга наиболее удачные моменты. Однако и сейчас каждый язык имеет массу неповторимых, только ему присущих особенностей, у каждого есть свои поклонники, предпочитающие именно его. Очевидно, в обозримом будущем языки будут существовать и развиваться параллельно, хотя *тенденция к сближению языков универсального назначения налицо.*

Перечислим наиболее популярные языки универсального назначения, соблюдая исторический порядок их появления и указывая их отличительные особенности. Более полный обзор современного состояния языков про-

граммирования можно найти в [20], где рассматриваются и редко используемые языки функционального и логического программирования (Lisp, Prolog и др.).

Fortran. Является первым из языков высокого уровня (1954—1958). В то время основной сферой применения вычислительных машин (термин "компьютер" вошел в обиход гораздо позже) были научно-технические расчеты. Для этой цели и был разработан простой и эффективный язык программирования. Базовые принципы, заложенные в язык Fortran, впоследствии легли в основу других языков программирования высокого уровня. Особенно сильно его влияние на Basic. В настоящее время популярность языка Fortran невелика, но его последние версии — Fortran 77 и Fortran 90 — продолжают использоваться в сфере научно-технических и инженерных расчетов.

Algol. Опубликован в 1960 г. От языка Fortran отличался значительно более строгими правилами синтаксиса, что позволило выявлять больше ошибок еще на этапе трансляции. Программы, написанные на этом языке, более понятны и выразительны, чем Fortran-программы. Благодаря этим качествам Algol нашел применение в научных кругах, в первую очередь среди специалистов по прикладной математике, теоретической и экспериментальной физике. Является непосредственным предшественником языка Pascal. Очевидно, язык Pascal оказался удачнее своего предшественника, поскольку практически его вытеснил.

Basic. Появился на свет в 1964 г. и был задуман как очень простой в изучении язык, который можно использовать как первый язык при обучении программированию. В настоящее время на его основе создан современный язык Visual Basic, используемый как простой и удобный язык широкого назначения, *но не в качестве языка для обучения программированию*. В этой роли он как-то не прижился.

Pascal. Язык разработан в 1967 г. швейцарским ученым Никлаусом Виртом *специально для целей обучения*. Автор несколько усовершенствовал правила языка Algol с целью удобства его освоения и применения, оставив неизменными достоинства — простоту и выразительность текста программы в сочетании с удобством отладки. В продвижении языка большая заслуга принадлежит специалистам фирмы Borland, которые разработали целую серию быстродействующих и эффективных трансляторов с Pascal и внесли ряд дополнений в сам язык.

Профессор Вирт не остановился на создании Pascal. На его основе им были разработаны два новых языка программирования — *Modula* (1970) и *Oberon* (1985), представляющие собой дальнейшее развитие Pascal. К сожалению, обстоятельства сложились так, что эти языки не приобрели той популярности, на которую можно было бы рассчитывать. Ученики Вирта сделали многое для популяризации и распространения языков Modula и особенно

Oberon. Ими был разработан новый диалект *языка* Oberon, названный *Component Pascal* (Компонентный Паскаль). Этим названием подчеркивается неразрывная связь всех языков на основе Pascal. Как в дальнейшем сложится судьба данного языка, пока сказать трудно.

Однако сам Pascal на протяжении многих лет был и остается не только *основным языком для обучения принципам программирования*, но и самым распространенным языком на многочисленных олимпиадах по программированию. На его основе фирмой Borland создана удобная среда разработки Delphi, пользующаяся большой популярностью, особенно в России.

Популярность языка Pascal в нашей стране столь велика, что его название обычно пишут по-русски — Паскаль. Однако авторы книги придерживаются англоязычного обозначения, поскольку названия многочисленных диалектов этого языка *(см. разд. 2.1.4)* не имеют устойчивых русскоязычных эквивалентов (Free Pascal, GNU Pascal, Object Pascal и т. д.).

C. Язык был создан в 1972 г. в компании Bell Laboratories. При разработке этого языка преследовались совсем другие цели, чем при создании языка Pascal. Язык C задумывался как *инструмент для разработки системного программного обеспечения*, т. е. как промежуточный между языками высокого и низкого уровня. От программ, написанных на этом языке, пытались добиться производительности, близкой к производительности программ на Ассемблере, но в то же время старались сохранить возможность переноса программ с одной *компьютерной платформы* на другую, что характерно для языков высокого уровня. При разработке синтаксиса основных конструкций языка преследовалась цель сделать текст программ как можно лаконичнее.

Разумеется, вопросы простоты изучения языка и удобства разработки программ на нем отошли на второй план, поскольку невозможно сразу удовлетворить всем требованиям. В связи с этим *при обучении программированию рекомендуется начинать с языка Pascal.* Хорошо усвоив основные принципы программирования на языке высокого уровня, можно затем по достоинству оценить лаконичность и эффективность программ, написанных на языке C, а само изучение этого языка уже не вызовет проблем.

C++. Объектно-ориентированная версия языка программирования C — C++ — была разработана в 1980 г. Бьорном Страуструпом в компании Bell Laboratories. Сегодня этот язык является одним из наиболее популярных среди профессионалов. Достаточно, например, сказать, что с его помощью была создана операционная система *Windows.*

Развитие языков C/C++ продолжается и в настоящее время. Появляются новые, молодые языки, авторы которых берут за основу наиболее удачные конструкции C/C++, убирая тяжеловесные или устаревшие элементы. Наиболее популярными современными языками программирования на основе C++ являются *Java* и *C#* (читается "Си шарп"). Оба эти языка не очень объемны, но достаточно стройны и удобны в освоении и использовании. К со-

жалению, программы на Java или C# пока уступают по скорости выполнения программам на C, что снижает возможности их использования в качестве языков системного программирования. Однако эти языки с успехом используются в сфере программирования для Internet.

Ada. Этот язык был разработан для создания программных систем с многолетним сроком службы и высокой степенью надежности. Ada создан по заказу и состоит на вооружении Министерства обороны США. На сегодняшний день этот язык считается одним из наиболее сложных, однако свои цели вполне оправдывает.

2.1.2. Состав языка программирования. Синтаксис и семантика

Обычный разговорный язык состоит из четырех основных элементов: символов (букв), слов, словосочетаний и предложений. Количество символов языка, образующих его алфавит, невелико. Количество слов неизмеримо больше, но все же конечно: все слова языка можно перечислить, например, сведя их в толковый словарь. Все словосочетания, а тем более предложения перечислить уже нельзя, но известны правила, по которым они составляются. Правила русского языка, например, изложены в соответствующих учебниках. *Аналогично устроены все языки программирования.*

Состав языка программирования

Язык программирования содержит элементы, перечисленные ниже.

- *Символы* — это основные неделимые знаки, из которых составляются все тексты программ на данном языке. Совокупность всех символов образует *алфавит языка*. Алфавит языка программирования несколько шире, чем алфавит естественного языка, и включает обычно латинские буквы, знаки арифметических операций, символы-разделители и ряд других специальных символов.
- *Лексемы* — это неделимые последовательности символов алфавита (элементарные конструкции), имеющие самостоятельный смысл. Они образуются из основных символов языка, так же как слова обычного языка строятся из букв. Возможны лексемы, состоящие из одного символа, например, знаки операций. Нельзя, однако, провести прямую аналогию между лексемами и словами обычного языка. Дело в том, ч*то любой язык программирования, конечно, имеет определенное количество *зарезервированных (ключевых) слов*, которые составляют *словарь* языка, но таких слов неизмеримо меньше, чем в естественном языке (всего несколько десятков). Однако программист может формировать по определенным правилам собственные слова — *идентификаторы*. Из-за этой особенности текст программы на языке высокого уровня воспринимается сложнее,

чем запись алгоритма на естественном языке. Совокупность лексем и правил их формирования образует *лексику языка*.

- *Выражения* строятся из лексем в строгом соответствии с правилами языка. Они задают порядок вычисления некоторого *значения*. Выражения играют в языке программирования ту же роль, что и словосочетания в обычном языке. Еще более близкий аналог выражений — математические формулы.
- *Операторы* (инструкции или команды языка) задают полное описание некоторого действия, которое необходимо выполнить. Это аналог предложения, выражающего законченную мысль, в обычном языке. Для описания сложного действия может потребоваться группа операторов. В этом случае операторы объединяются в *составной оператор* или *блок*.

Действия, заданные операторами, выполняются над *данными*. Предложения языка, в которых даются сведения о данных, называются *описаниями* или *неисполняемыми операторами*.

Совокупность описаний и операторов языка программирования, реализующая алгоритм решения конкретной задачи, образует программу на данном языке.

Синтаксис и семантика языка

Система правил записи элементов языка программирования (лексем, выражений, операторов) образует его *синтаксис*.

Смысл конструкций языка — это его *семантика*.

Иными словами, синтаксис языка определяет внешний вид отдельных конструкций ("как они выглядят"), а семантика толкует действия, которые выполняет компьютер по этим конструкциям ("что они делают").

Современные языки высокого уровня очень схожи по своей семантике, но имеют некоторые непринципиальные различия в синтаксисе. Отсюда проистекает важный вывод: *начинающим изучать программирование следует обратить внимание именно на семантику языка.* Именно она представляет собой те *базовые знания*, которые не зависят от особенностей конкретного языка. Хорошее знание основ поможет в дальнейшем изучать новые языки программирования в короткие сроки и при этом писать программы хорошего качества независимо от синтаксиса языка.

Справедливости ради следует признать, что изучение правил синтаксиса языка обычно не вызывает затруднений. Понять до конца семантику каждой конструкции гораздо труднее. Еще труднее научиться использовать каждую конструкцию языка по назначению при записи своих собственных алгоритмов. К сожалению (или к счастью), тут можно предложить только один старый испытанный способ — упорную работу за компьютером, разбор многочисленных примеров, решение задач.

2.1.3. Описание языка

В отличие от естественных языков, для каждого языка программирования имеется его строгое описание. Зачем нужно такое описание, очевидно понятно, ведь речь идет о языке, предназначенном для общения с компьютером, где малейшая двусмысленность приводит к ошибкам.

Описание языка есть описание его основных элементов, причем описание каждого элемента языка задается его синтаксисом и семантикой. Синтаксические определения устанавливают правила построения элементов языка. Семантика определяет смысл и правила использования тех элементов языка, для которых были даны синтаксические определения.

Неформальное описание языка

Для описания языка программирования можно использовать обычный естественный язык. Но такое описание (назовем его неформальным) будет страдать теми же недостатками, что и описание какого-либо алгоритма в словесной форме: излишним многословием при опасности неоднозначного толкования. Правда, в отличие от описания алгоритма, описание языка рассчитано на изучение его человеком (программистом), поэтому в литературе по программированию такое описание применяется довольно часто.

Однако разработчики трансляторов (это тоже программисты, но не *прикладные*, а *системные*) предпочитают иметь более формальное описание языка, т. е. такое, которое не допускает никакого неоднозначного толкования и при этом является компактным и удобным в использовании для профессионалов. В соответствии с их пожеланиями уже для описания одного из первых языков (Algol) был придуман специальный сугубо формальный язык описания.

Способы формального описания языка

Сейчас имеется несколько языков описания. Для того чтобы отличать их от языков программирования, которые они описывают, их называют *метаязыками*.

Дадим представление о двух разных метаязыках, которые используются для одних и тех же целей и являются полностью взаимозаменяемыми.

- Формы Бэкуса—Наура (БНФ). Именно этот метаязык был предложен для описания языка Algol, а затем многократно использовался авторами различных языков программирования для формального описания их творений. В настоящее время в основном используется расширенный вариант, который называется РБНФ (расширенные БНФ).
- Синтаксические диаграммы Вирта. Графическое изображение основных конструкций языка.

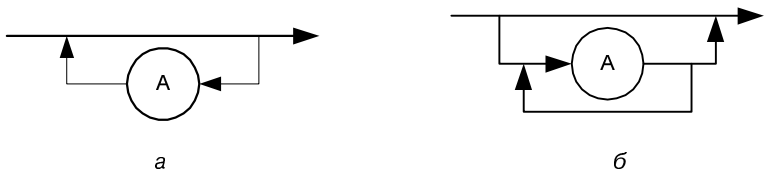
Формы Бэкуса—Наура позволяют очень строго и компактно описать все элементы языка, используя несколько символов специального назначения — *метасимволы*. При этом каждое понятие языка выводится из более простых понятий.

Самые простые понятия — это символы и зарезервированные слова языка. Они называются *терминальными символами* или *терминалами*. В описаниях терминалы берутся в кавычки или выделяются жирным шрифтом.

Все остальные понятия — *нетерминалы* — постепенно выводятся из терминальных с помощью специальных металингвистических формул, в которых вместо знака равенства используется знак `::=`, читаемый как "есть по определению". Каждый нетерминал заключается в угловые скобки. Метасимвол `|` имеет обычное значение "или".

Расширенная БНФ дополнена еще несколькими полезными метасимволами. Так, запись `{A}` обозначает повторение символа `A` несколько (возможно и нуль) раз. Запись `[A]` обозначает повторение символа `A` нуль или один раз (таким образом удобно обозначать необязательные элементы языка). Кроме того, формулы РБНФ выглядят несколько проще, чем в БНФ, т. к. понятия не берутся в угловые скобки, а знак `::=` заменен на обычное равенство. Терминальные символы берутся в кавычки.

Синтаксические диаграммы Вирта — это графическое описание языка с использованием блоков, соединенных линиями. При этом терминалы обозначаются при помощи кружков (или овалов), а нетерминалы — при помощи прямоугольников. Для каждой конструкции Бэкуса—Наура можно построить эквивалентную синтаксическую диаграмму. Например, конструкциям `{A}` и `[A]` соответствуют синтаксические диаграммы, изображенные на рис. 2.1, *а*, *б*.



а *б*

Рис. 2.1. Синтаксические диаграммы конструкций `{A}` и `[A]`

Приведем небольшой пример, иллюстрирующий применение этих способов. Пусть необходимо описать такую несложную конструкцию любого языка программирования, как целое число.

Описание с использованием БНФ имеет вид:

```
<Целое>::=<ЦелоеБезЗнака>|<Знак><ЦелоеБезЗнака>
<ЦелоеБезЗнака>::=<Цифра>|<ЦелоеБезЗнака><Цифра>
```

```
<Знак>::=+|-
<Цифра>::=0|1|2|3|4|5|6|7|8|9.
С использованием РБНФ:
Целое=[Знак]Цифра{Цифра}
Знак= "+"|"-"
Цифра="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
```

Соответствующая синтаксическая диаграмма приведена на рис. 2.2.

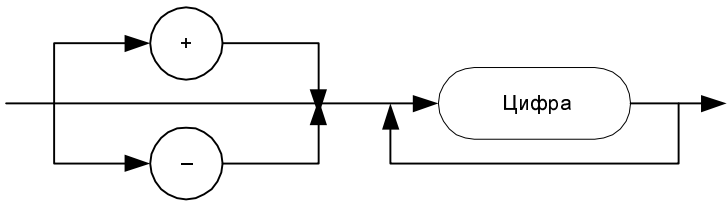


Рис. 2.2. Синтаксическая диаграмма описания целого числа

Неформальное описание этой же конструкции: целое число — это последовательность цифр, которой может предшествовать знак.

Несмотря на то, что неформальное описание — наименее строгое из всех способов описания языка, в данной книге будет использоваться именно этот способ. На это есть, по крайней мере, две причины:

- неформальное описание легче воспринимается начинающими. С целым числом, конечно, все понятно при любом способе описания, но чтение более сложных конструкций требует определенных усилий;
- при неформальном описании можно сделать упор именно на смысл конструкций языка. Вспомним, что основная цель при изучении первого языка программирования — как можно лучше понять его семантику.

Возможные неточности неформального описания будут уточняться с помощью большого количества примеров, для того чтобы полностью исключить возможность неоднозначного толкования.

Однако пример формальной записи более серьезной конструкции языка (арифметического выражения) в книге все же имеется *(см. разд. 14.4.4)*.

2.1.4. Стандарт и реализации языка

Естественные языки создавались веками, поэтому они довольно консервативны. Но определенные изменения с течением времени все же происходят: появляются новые слова и выражения, исчезают устаревшие обороты. Время