А. В. Могилев
Л. В. Листрова

МЕТОДЫ ПРОГРАММИРОВАНИЯ КОМПЬЮТЕРНЫЕ ВЫЧИСЛЕНИЯ

Санкт-Петербург
«БХВ-Петербург»
2008

УДК 681.3.06(075.3)
ББК 32.973.26-018.2я721
М74

Могилев, А. В.

М74 Методы программирования. Компьютерные вычисления / А. В. Могилев, Л. В. Листрова. — СПб.: БХВ-Петербург, 2008. — 320 с.: ил. — (ИиИКТ)

ISBN 978-5-9775-0151-4

Книга является частью комплекта учебников по курсу информатики и информационно-коммуникационных технологий (ИКТ) в старших классах общеобразовательной школы на профильном уровне. Она охватывает 5-й и 6-й из 10-ти модулей курса и является продолжением пособий "Информация и информационные процессы. Социальная информатика", "Средства информатизации. Телекоммуникационные технологии".

В книге рассмотрены история развития языков программирования и парадигмы программирования, языки программирования высокого уровня, метаязыки для описания синтаксических конструкций языка высокого уровня, структурно-ориентированное программирование и язык Паскаль, введение в язык Си, элементы объектного программирования, основы логического программирования на языке Пролог, вычислительные методы, дано понятие о компьютерном моделировании.

По каждой рассматриваемой теме есть контрольные вопросы, темы для рефератов и докладов, вопросы для обсуждения, задачи и упражнения, лабораторные работы.

Для учащихся старших классов физико-математического, информационно-технологического и других профилей

УДК 681.3.06(075.3)
ББК 32.973.26-018.2я721

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Антонина Панюшева*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Людмила Минина*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.07.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 25,8.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0151-4

© Могилев А. В., Листрова Л. В., 2008
© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

Предисловие

Несмотря на значительные усилия, направленные на становление профильного образования, изменения в российской школе еще далеки от ожиданий. Одна из причин связана с недостаточностью учебно-методического обеспечения обучения на профильном уровне во многих образовательных областях, к которым относятся также информатика и информационные технологии.

Данный цикл пособий призван хотя бы отчасти снизить остроту этой проблемы и ликвидировать пробел в учебной литературе, предназначенной для обучения старшеклассников по профилям, включающим курс информатики и информационно-коммуникационных технологий профильного уровня: физико-математическом, информационно-технологическом и др.

Мы предлагаем модульный вариант как самого курса информатики и информационно-коммуникационных технологий, так и его учебного обеспечения. Значительный объем всего профильного курса (280 часов), разнородность его содержательных компонент (как с точки зрения соотношения теоретических и практических вопросов, так и сложности и трудоемкости деятельности учащихся), быстрое и неоднородное их развитие диктуют именно модульное построение данного курса, позволяющее выбирать, использовать и развивать модули учебного пособия независимо друг от друга. Кроме того, модульная структура курса обеспечивает большую гибкость и эффективность в достижении качества обучения, поскольку оно гарантируется качеством освоения каждого модуля в отдельности.

Согласно авторской учебной программе данный профильный учебный курс состоит из следующих 10 модулей:

1. Информация и информационные процессы.
2. Социальная информатика.
3. Средства информатизации.
4. Телекоммуникационные технологии.
5. Методы программирования.
6. Компьютерные вычисления.

7. Технологии обработки текстовой информации.
8. Технологии обработки графической и мультимедийной информации.
9. Технологии поиска и хранения информации.
10. Технологии автоматизации управления.

Настоящее пособие охватывает 5-й и 6-й модули курса и является продолжением пособий "Информация и информационные процессы. Социальная информатика", "Средства информатизации. Телекоммуникационные технологии" тех же авторов.

Пособия содержат учебный материал и различного рода задания и вопросы: тексты для чтения и изучения, контрольные вопросы, темы рефератов, вопросы для обдумывания и обсуждения, задания и упражнения, а также описание лабораторных работ. Это позволяет организовать занятия различного типа и реализовывать различные методы обучения от лекций и семинарских занятий до учебных проектов.

Для того чтобы облегчить ориентацию в пособии и придать ему структуру, мы используем следующие обозначения:

-  — **Учебный материал**;
-  — **Контрольные вопросы**;
-  — **Темы для рефератов и докладов**;
-  — **Вопросы для обсуждения**;
-  — **Задачи и упражнения**;
-  — **Лабораторные работы**.

МОДУЛЬ 5

Языки и методы программирования

5.1. История развития языков программирования и парадигмы программирования

Учебный материал

Основоположником программирования можно считать английского математика Чарлза Бэббиджа (1791—1871 гг.). В 20-х годах XIX века ему пришла идея создать такую механическую машину для вычислений, что порядок ее действий можно было предварительно записывать и впоследствии выполнять эти действия на машине автоматически. Это была идея, положившая начало программированию. Ч. Бэббидж посвятил реализации этой идеи всю жизнь. Он не добился успеха и признания современников при жизни, но оказал огромное влияние на современное развитие информатики.

Хотя использованный Бэббиджем способ записи программы на перфокартах, придуманный для управления ткацкими станками французским изобретателем Жозефом Мари Жаккаром, не имеет ничего общего с современными технологиями хранения и выполнения программ компьютерами, принцип остался тем же.

Рядом с Ч. Бэббиджем у истоков программирования стояла Ада Лавлейс, дочь английского поэта Чарлза Гордона Байрона. Она оказалась одним из немногих современников Чарлза Бэббиджа, кто сумел по достоинству оценить идею "аналитической машины". Она стала ближайшей помощницей и сподвижницей Бэббиджа, разработала некоторые приемы управления последовательностью вычислений, которые используются в программировании и по сей день, описала одну из важнейших конструкций практически любого современного языка программирования — цикл. Аду Лавлейс по праву считают первым в мире программистом.

Дальнейшего прогресса в программировании пришлось ждать почти 100 лет, и связан он был с появлением в середине 1940-х годов электромеханических

и электронных вычислительных машин — родителей современных компьютеров.

Для программирования этих машин использовались машинные коды — цифровые комбинации, "понятные" только данной машине. Такое программирование было чрезвычайно трудоемким и сложным делом, доступным лишь небольшому кругу специалистов.

Первым шагом в развитии современных языков программирования стало создание в конце 1940-х годов Джоном Моучли, сотрудником Пенсильванского университета (США), системы кодирования машинных команд этих компьютеров с помощью специальных символов. Достижением создателей языков программирования было то, что они сумели заставить сам компьютер работать переводчиком с этих языков на машинный код. Описываемая система, которую назвали "Short Code", была по существу одним из первых примитивных интерпретаторов. Она использовала примитивный язык программирования высокого уровня. На нем программист записывал решаемую задачу в виде математических формул, а затем сам, используя специальную таблицу, переводил символ за символом, преобразовывая эти формулы в двухлитерные коды. В дальнейшем специальная программа компьютера превращала эти коды в двоичный машинный код.

Система кодирования, предложенная Моучли, увлекла одну из сотрудниц его группы — Грейс Мюррей Хоппер, которая стала третьим в мире программистом.

В 1951 г. Хоппер создала первый в мире компилятор. Именно она ввела сам этот термин. Компилятор Хоппер осуществлял функцию объединения команд и в ходе трансляции производил организацию подпрограмм, выделение памяти компьютера, преобразование команд высокого уровня (в то время псевдокодов) в машинные команды. "Подпрограммы находятся в библиотеке (компьютера), а когда вы подбираете материал из библиотеки — это называется компиляцией", — так она объясняла происхождение введенного ею термина.

В 1954 г. группа под руководством Г. Хоппер разработала систему, включающую язык программирования и компилятор, которая в дальнейшем получила название MATH-MATIC. После удачного завершения работ по созданию MATH-MATIC Г. Хоппер и ее группа принялись за разработку нового языка и компилятора, который позволил бы пользователям программировать на языке, близком к обычному английскому. В 1958 г. появился компилятор FLOW-MATIC. FLOW-MATIC был первым языком для задач обработки коммерческих данных. Работы в этом направлении привели к созданию языка КОБОЛ (COBOL — Common Business Oriented Language). Одним из основных консультантов при его создании была Грейс Мюррей Хоппер.

При работе на компьютере "Марк-I" Г. Хоппер и ее группе пришлось стать первопроходцами программирования. Они первыми придумали подпрограммы. Сейчас любой программист не задумываясь использует подпрограммы в любом языке программирования. Еще одно фундаментальное понятие техники программирования этой группы — "отладка". Однажды жарким летним днем 1945 г. неожиданно произошла остановка компьютера "Марк-I". Обнаружилась неисправность одного реле, контакты которого были заблокированы мотыльком, залетевшим в помещение вычислительного центра. Г. Хоппер вспоминала: "Когда к нам зашел офицер, чтобы узнать, чем мы занимаемся, мы ответили, что удаляем из компьютера насекомых (debuging)". С тех пор термин "debuging" (отладка) используется в технических процессах тестирования неисправностей в компьютере, а также в системах программирования.

Середина 50-х годов XX века характеризуется стремительным прогрессом в области программирования. Программирование в машинных командах стало вытесняться программированием на языках, выступавших в роли посредника между машинами и программистами. Первым и одним из наиболее распространенных стал Фортран (FORTRAN, от FORmula TRANslator — переводчик формул), разработанный группой программистов фирмы IBM в 1954 г. Этот язык получил большое распространение, стал основным языком для научных и технических расчетов, несколько раз усовершенствовался и широко используется до сих пор.

В конце 50-х годов плодом международного сотрудничества в области программирования явился Алгол-60 (ALGOL, от ALGOrithmic Language — алгоритмический язык, версия 1960 г.). Алгол предназначен для записи алгоритмов, которые строятся в виде последовательности процедур, применяемых для решения поставленных задач. Специалисты-практики восприняли этот язык далеко не однозначно, но, тем не менее, его влияние на развитие других языков и теорию программирования оказалось весьма значительным.

Развитие идеи Алгола о структуризации разработки алгоритмов нашло наивысшее отражение при создании в начале 1970-х годов языка Паскаль швейцарским ученым Никлаусом Виртом. Язык Паскаль первоначально разрабатывался как учебный, и, действительно, сейчас он является одним из основных языков обучения программированию в школах и вузах. Однако качества его в совокупности оказались столь высоки, что им охотно пользуются и профессиональные программисты.

В середине 1960-х годов сотрудники математического факультета Дартмутского колледжа Томас Курц и Джон Кемени создали специализированный язык программирования, который состоял из простых слов английского языка. Новый язык назвали "универсальным символическим кодом для начинающих" (Beginners All Purpose Symbolic Instruction Code, или, сокращенно, BASIC, а по-русски — Бейсик). Годом рождения нового языка можно считать 1964 г.

Долгое время универсальный язык Бейсик (имеющий множество версий) имел большую популярность и широкое распространение среди пользователей ЭВМ различных категорий во всем мире. В значительной мере этому способствовало то, что Бейсик начали использовать как встроенный язык персональных компьютеров, широкое распространение которых началось в конце 1970-х годов.

Большой отпечаток на современное программирование наложил язык Си (первая версия — 1972 г.), являющийся очень популярным в среде разработчиков систем программного обеспечения (включая операционные системы). Си сочетает в себе черты как языка высокого уровня, так и машинно-ориентированного языка, допуская программиста ко всем машинным ресурсам, чего не позволяют такие языки, как Бейсик и Паскаль.

Более специализированным языком является язык ЛОГО (от греческого logos — слово), созданный для обучения программированию школьников профессором математики и педагогики Сеймуром Пейпертом из Массачусетского технологического института. Обучаясь программированию на ЛОГО, дети задают простые команды и составляют из них программы, которые управляют условной "черепашкой" — объектом, оставляющем при перемещении след на экране монитора.

Отметим язык LISP (LISt Processing — обработка списков), появившийся в США в конце 1950-х годов, и еще один специализированный язык — Пролог (Prolog — PROgramming in LOGic), разработанный в 1970-е годы, как языки программирования для создания систем искусственного интеллекта.

В начале 1960-х годов все существующие языки программирования высокого уровня можно было пересчитать по пальцам, однако впоследствии их число достигло трех тысяч. Однако в практической деятельности используется не более двух десятков из них.

Разработчики ориентировали языки на разные классы задач, в той или иной мере привязывали их к конкретным архитектурам компьютеров, воплощали в них личные вкусы и идеи.

В конце 1960-х годов были сделаны попытки преодолеть эту "разноголосицу" путем создания универсального языка программирования. Первым детищем этого направления стал PL/1 (Programm Language One), 1967 г. Затем на эту роль претендовал Алгол-68 (1968 г.). Предполагалось, что подобные языки будут развиваться и усовершенствоваться и вытеснят все остальные. Однако ни одна из этих попыток на сегодняшний день не увенчалась успехом (хотя PL/1 в усеченных версиях использовали многие программисты). Стремление к универсальности языка приводило к неоправданной сложности конструкций программы, неэффективности получаемых исполняемых кодов.

Бурный рост числа различных языков программирования в период с конца 1960-х и до начала 1980-х годов сопровождался, как это ни странно, кризисом программного обеспечения. Остро не хватало программ для решения самых разных задач и программистов для их разработки, а написанные программы часто содержали ошибки и работали неустойчиво. Этот кризис особо остро переживало военное ведомство США. В январе 1975 г. Пентагон решил навести порядок в хаосе трансляторов и учредил комитет, которому было предписано разработать один универсальный язык. На конкурсной основе комитет проработал сотни проектов, и когда стало ясно, что ни один из существующих языков не может их удовлетворить, принял два проекта для окончательного рассмотрения. В мае 1979 г. был объявлен победитель — группа ученых во главе с Жаном Ихбиа. Победивший язык окрестили АДА, в честь Ады Лавлейс. Язык АДА — прямой наследник языка Паскаль. Этот язык предназначен для создания и длительного (многолетнего) сопровождения больших программных систем, допускает возможность параллельной обработки, управления процессами в реальном времени и многое другое, чего трудно или невозможно достичь средствами более простых языков.

Следует отметить, что многие языки, первоначально разработанные для больших и малых вычислительных машин, в дальнейшем были приспособлены к персональным компьютерам.

Языки программирования сохраняют свое предназначение для решения задач определенных типов. Выбор языка определяется удобствами для программистов, их предпочтениями в силу опыта и образования, а также пригодностью для данного компьютера и данной задачи. А задачи, решаемые с помощью компьютера, бывают самые разнообразные: вычислительные, экономические, графические, экспертные, создание системного программного обеспечения и т. д. Такая разнотипность решаемых компьютером задач и приводит к многообразию языков программирования.

Наилучший результат в программировании достигается при индивидуальном выборе языка программирования на основе класса задачи, уровня и интересов программиста. Например, Бейсик широко употребляется при обучении и написании простейших программ; Фортран является классическим языком программирования при решении математических и инженерных задач; Кобол используется как основной язык для массовой обработки данных в сферах управления и бизнеса.

Несмотря на многообразие языков программирования, каждый из них можно отнести к одной из всего трех парадигм. *Парадигмой* в науковедении называется набор теорий, гипотез, взглядов и подходов, относящихся к одному течению, в основе которых лежит общий принцип. В программировании известны три парадигмы: процедурная, функциональная и логическая. Боль-

шинство языков программирования, доминирующих при создании системного и прикладного программного обеспечения, таких как Фортран, Бейсик, Паскаль, Ада, Си, Модула, Форт, относятся к процедурной парадигме.

Однако по мере эволюции языков программирования получили широкое распространение и другие, принципиально иные, подходы к созданию программ.

Сущность процедурного программирования (его также называют операциональным, классическим) состоит в детальном описании шагов, действий, которые должен выполнить компьютер для того, чтобы решить задачу. Другими словами, процедурное программирование — это запись алгоритма средствами языка программирования. При этом ожидаемые свойства результата обычно не указываются. Основные понятия языков этих групп — оператор и данные. При процедурном подходе операторы объединяются в процедуры.

Внутри процедурной парадигмы можно выделить *структурное программирование*, которое не выходит за рамки этого направления, оно лишь использует полезные приемы программирования (структурная запись программы, отказ от оператора перехода, составные операторы или блоки), что делает программистскую деятельность более производительной и защищенной от ошибок и путаницы в программах.

Принципиально иные направления в программировании относятся к непроцедурным парадигмам. К ним можно отнести *объектно-ориентированное* и *декларативное программирование*. Из языков объектного программирования, имеющих популярность, следует назвать Си++, среды типа Delphi и Visual Basic.

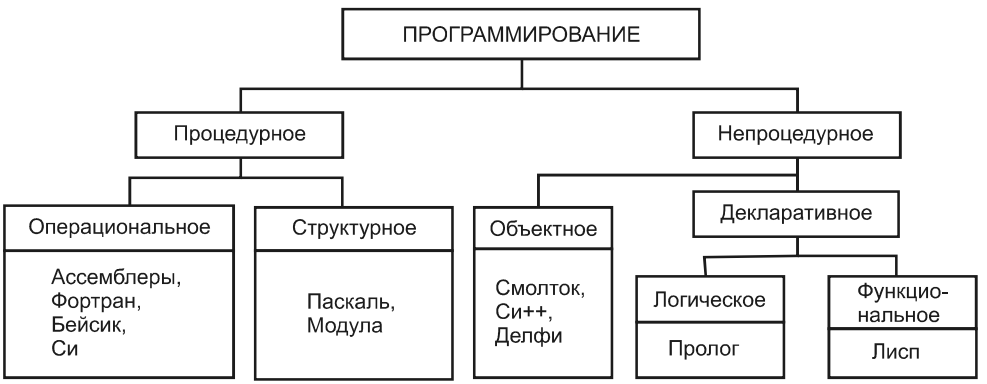


Рис. 5.1. Классификация языков программирования

При использовании декларативного языка программист задает исходные информационные структуры, взаимосвязи между ними и то, какими свойствами должен обладать результат. При этом процедуру его получения ("алгоритм")

программист не описывает. В такого рода языках отсутствует понятие "оператор" ("команда"). Декларативные языки можно подразделить на два семейства: логические (типичный представитель — Пролог) и функциональные (Лисп).

По всей видимости, непроцедурные языки имеют большое будущее.

Общая классификация языков программирования в соответствии с парадигмами программирования приведена на рис. 5.1. Изучая этот курс, вы познакомитесь с несколькими языками программирования, относящимися к разным парадигмам.

Контрольные вопросы

1. Как зарождалось программирование в докомпьютерную эпоху?
2. Каков вклад в появление первых языков программирования высокого уровня Дж. Моучли и Г. Хоппер?
3. Охарактеризуйте назначение языков Фортран, Алгол, Си, Паскаль.
4. В чем причина неудач универсальных языков PL/1 и Алгол-68?
5. Какие языки используются для разработки систем искусственного интеллекта?
6. Какие языки ориентированы на задачи обучения программированию?
7. Что называется парадигмой программирования?
8. Приведите примеры языков программирования, относящихся к различным парадигмам.

Темы для рефератов и докладов

1. Технологии программирования первых компьютеров.
2. Развитие машинных языков программирования.
3. Языки для научных вычислений.
4. Языки обработки символьной информации.
5. Языки разработки систем искусственного интеллекта.

6. Языки для начального обучения программированию.
7. Языки объектного программирования.

Вопросы для обсуждения

1. Почему количество языков программирования так велико?
2. Может ли программирование стать ненужным?
3. Почему изучение языков программирования вызывает затруднения?

Задачи и упражнения

1. Составьте сводную справочную таблицу языков программирования. Отразите в ней название языка (или его версии), год создания, назначение, парадигму, к которой язык относится.

Лабораторные работы

1. Выясните, какие книги имеются в библиотеке вашего учебного заведения (и других доступных библиотеках), посвященные языкам и технологиям программирования. Какие это языки? Составьте сводную таблицу-каталог.
2. Выполните поиск в Интернете ресурсов, посвященных языкам и технологиям программирования. Постарайтесь оценить эти ресурсы на предмет доступности. Составьте каталог этих ресурсов.
3. Выясните, какие системы программирования и на каких языках доступны в вашем учебном заведении.

5.2. Языки программирования высокого уровня. Метаязыки для описания синтаксических конструкций языка высокого уровня

Учебный материал

Языки программирования — это формальные языки, специально созданные для общения человека с компьютером.

Языками *высокого уровня* называют языки программирования, универсальные по отношению к архитектуре компьютеров и использующие обозначения, близкие к принятым в математике и других видах деятельности человека. На таких языках удобно писать прикладные программы, решающие какие-то научные, технические, управленческие задачи. Программы на языках высокого уровня содержат служебные слова естественного (английского) языка, состоят из легко читаемых и понимаемых команд. Это отличает такие языки от языков *низкого уровня* (машинно-ориентированных).

Машинно-ориентированные языки содержат примитивные команды, соответствующие особенностям данной архитектуры компьютера (в этом и состоит их машинная ориентированность), и к тому же записываемые машинными кодами, обычно в шестнадцатеричной форме, типа: переслать число в ячейку; считать число из ячейки; увеличить содержимое ячейки на +1 и т. п. Команда на машинном языке обычно описывает простейший обмен содержимого ячеек памяти, элементарные арифметические и логические операции. Команда содержит код и адреса ячеек, с содержимым которых выполняется закодированное действие.

С машинно-ориентированными языками трудно работать, но созданные с их помощью квалифицированным программистом программы занимают меньше места в памяти и работают быстрее. С помощью этих языков удобнее разрабатывать системные утилиты, антивирусные программы, драйверы (программы для управления устройствами компьютера), некоторые другие виды программ.

Языки программирования высокого уровня имеют следующие достоинства:

- ❒ алфавит языка значительно шире машинного, что делает его гораздо более выразительным и существенно повышает наглядность и понятность текста;
- ❒ набор операций, допустимых для использования, не зависит от набора машинных команд, а выбирается из соображений удобства записи алгоритмов решения задач определенного класса;
- ❒ конструкции команд (операторов) отражают привычные человеку приемы обработки данных и задаются в удобном для чтения и понимания виде;
- ❒ используется аппарат переменных и действия с ними;
- ❒ поддерживается широкий набор типов данных.

Языки программирования высокого уровня являются машинно-независимыми и требуют использования соответствующих программ-переводчиков (*трансляторов*) для перевода программы на язык компьютера, на котором она будет исполняться.

Каждый язык программирования, равно как и "естественный" язык (русский, английский и т. д.), имеет алфавит, словарный запас, свои грамматику и синтаксис, а также семантику.

Алфавит — фиксированный для данного языка набор основных символов, допускаемых для составления текста программы на этом языке.

Синтаксис — система правил, определяющих допустимые конструкции языка программирования из букв алфавита.

Семантика — система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести процесс обработки данных.

Понятие подразумевает некоторую синтаксическую конструкцию и определяемые ею свойства программных объектов или процесса обработки данных.

Взаимодействие синтаксических и семантических правил определяют те или иные понятия языка, например операторы, идентификаторы, переменные, функции и процедуры, модули и т. д. В отличие от естественных языков, правила грамматики и семантики для языков программирования, как и для всех формальных языков, должны быть явно, однозначно и четко сформулированы.

Для строгого и точного описания синтаксиса языка программирования, как правило, используют специальные *метаязыки* (языки для описания других языков). Наиболее распространенными метаязыками являются *металингвистические формулы Бэкуса — Наура* (язык БНФ) и *синтаксические диаграммы Вирта*.

Язык БНФ (называемый также языком нормальных форм) представляет собой способ записи конструкций языка программирования с помощью формул, похожих на математические. Для каждого понятия языка существует единственная метаформула (нормальная форма). Она состоит из левой и правой частей. В левой части указывается определяемое понятие, а в правой — задается множество допустимых конструкций языка, которые объединяются в это понятие. В формуле используют специальные метасимволы в виде угловых скобок, которые обозначают определяемое понятие (в левой части формулы) или ранее определенное понятие (в ее правой части). Левая и правая части формулы разделяются метасимволом "::=", имеющим смысл "по определению есть". Знак "|" следует читать "или".

Например, метаформулы

```
<переменная>::=A|B
<выражение>::=<переменная>|<переменная>+<переменная>|<переменная>-
<переменная>
```

означают, что <переменная> это одна из букв, A или B, а <выражение> — любая из следующих десяти записей: A; B; A+A; A+B; B+A; B+B; A–A; A–B; B–A; B–B.

Правая часть метаформулы может содержать правило построения допустимых последовательностей. Допускаются рекурсивные определения терминов и понятий, т. е. когда в правой части формулы участвует понятие, определяемое левой частью. Например, пусть необходимо ввести понятие <двоичный код>, под которым понимается любая непустая последовательность цифр 0 и 1. Тогда простое и компактное рекурсивное определение с помощью метаформул выглядит так:

```
<двоичная цифра>::= 0 | 1
<двоичный код>::=<двоичная цифра>|<двоичная цифра><двоичный код>
```

Данное определение позволяет определить, является ли некая конструкция определяемым понятием.

Так, для нашего примера метаформулы двоичного кода конструкция 001101 является двоичным кодом, поскольку, последовательно применяя рекурсию, мы имеем следующие 5 шагов рекурсии:

1. 0 — двоичная цифра, а 01101 — двоичный код, поскольку
2. 1 — двоичная цифра, а 1101 — двоичный код, поскольку
3. 1 — двоичная цифра, а 101 — двоичный код, поскольку
4. 1 — двоичная цифра, а 01 — двоичный код, поскольку
5. 0 — двоичная цифра, а 1 — также двоичная цифра, рекурсия завершена.

Если же взять конструкцию 1021, то она не является двоичным кодом, поскольку:

1. 1 — двоичная цифра, а 021 — не двоичный код, поскольку
2. 0 — двоичная цифра, а 21 — не двоичный код, поскольку
3. 2 — не двоичная цифра.

Для задания синтаксических конструкций произвольной длины часто используют фигурные скобки как метасимволы. Фигурные скобки означают, что конструкция может повторяться нуль или более раз. В частности, термин <двоичный код> можно определить по-другому, а именно:

```
<двоичный код>::=<двоичная цифра>{<двоичная цифра>}
```

И еще, для полноты множества синтаксических конструкций, необходимо определить конструкцию <пусто>:

```
<пусто>::= .
```

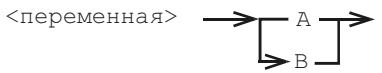
В большинстве учебных пособий по программированию, технических описаний языков, метаформулы рассматриваемого языка представлены полностью.

Синтаксические диаграммы позволяют графически отобразить значения метапеременных метаязыка. Диаграмма состоит из основных символов или понятий языка.

Каждая диаграмма имеет входящую и выходящую стрелки, означающие начало и конец синтаксической конструкции и отражающие процесс ее чтения и анализа. Из каждого элемента выходит одна или несколько стрелок, указывающих на те элементы, которые могут следовать непосредственно за данным элементом.

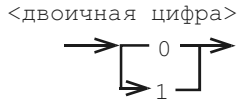
Для сравнения с метаформулами приведем несколько примеров.

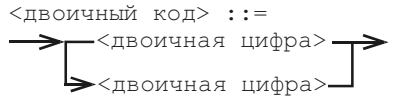
Синтаксическая диаграмма



эквивалентна метаформуле <переменная>::= A|B.

Еще примеры:





Металингвистические формулы заложены в трансляторы; с их помощью ведется проверка используемых программистом конструкций на формальное

соответствие какой-нибудь из конструкций, синтаксически допустимых в этом языке (синтаксический контроль).

Пользуясь средствами форм Бэкуса — Наура и синтаксическими диаграммами, опишем грамматику языков программирования.

Алфавиты большинства языков программирования близки друг другу и основываются на буквах латинского алфавита, арабских цифрах и общепринятых спецсимволах, таких как знаки препинания, математических операций, сравнений и обозначений. Большинство популярных языков программирования в своем алфавите содержат следующие элементы:

```
<буква> : : =  A|a|B|b|C|c|D|d|E|e|F|f и т. д.

<цифра> : : = 0|1|2|3|4|5|6|7|8|9

<знак арифметической операции > : : = *|/|+|-

<разделитель> : : = . | , | ; | : | ( | ) | [ | ] |{ | }| ' | : =

<служебное слово> : : = begin | end | if | then | else| for | next и
т. д.

<спецсимвол> :: = <знак арифметической операции> | <разделитель> |
<служебное слово>

<основной символ>::=<буква> | <цифра> | <спецсимвол>

<комментарий>::= {любая последовательность символов}
```

Несмотря на значительные различия между языками программирования, ряд фундаментальных понятий в большинстве из них схожи.

Оператор — одно из главных понятий всех языков программирования. Каждый оператор представляет собой законченную фразу языка и однозначно определяет некоторое действие над данными. В соответствии с теорией алгоритмов выделяют основные (базовые) операторы языка:

- ❒ присвоения;
- ❒ условного и безусловного перехода;
- ❒ пустой оператор.

К производным операторам относят:

- ❒ составной оператор;
- ❒ оператор выбора;

- оператор цикла;
- оператор присоединения.

Все операторы языка в тексте программы отделяются друг от друга явными или неявными разделителями, например:

```
S1; S2; ...; Sn
```

Операторы выполняются в порядке их следования в тексте программы. Лишь с помощью операторов перехода этот естественный порядок может быть нарушен.

Большая часть операторов ведет обработку величин.

Величины могут быть *постоянными* и *переменными*. Значения постоянных величин не изменяются в ходе выполнения программы. Величина характеризуется *типом*, *именем* и *значением*. Наиболее распространенные типы величин — числовые (целые и вещественные), символьные, логические. Тип величины определяется ее значением.

Другая важная классификация величин — *простые* и *составные*. Простая величина имеет одно значение. Ей соответствует одна ячейка памяти (точнее, "машинное слово") или ее эквивалент во внешней памяти компьютера.

Составной величине может соответствовать сразу много значений, объединенных под одним именем. Эти значения представляют собой элементы (компоненты) величины. Самый широко известный пример — массив, у которого элементы различаются по индексам (номерам).

Вопрос о выборе и описании структур данных — входных, выходных и промежуточных — не менее важен для успеха решения прикладной задачи, чем вопрос о правильности последовательности операторов программы.

Важнейшие характеристики структурированной величины таковы: *упорядоченность* (да или нет), *однородность* (да или нет), *способ доступа* к элементам, *постоянство числа элементов* (да или нет). Так, массив является упорядоченной однородной структурой с прямым доступом к элементам и постоянным их количеством.

Всем объектам данных в языках программирования даются индивидуальные **имена**. Имя программного объекта называют *идентификатором* (от слова "идентифицировать"). Чаще всего идентификатором является любая конечная последовательность букв и цифр, начинающаяся с буквы:

```
<идентификатор>::=<буква>
<идентификатор> ::= <идентификатор>(<буква>| <цифра>)
```

Как правило, в качестве идентификатора запрещается использовать служебные слова языка.

Многим слово "идентификатор" не нравится, и в настоящее время чаще употребляют слово "имя", поскольку

```
<имя>::=<идентификатор>
```

Программисты выбирают имена по своему усмотрению. Принципы выбора и назначения имен объектам данных естественны. Следует избегать мало выразительных обозначений, а также кратких имен. Имена должны быть понятны, наглядны, отражать суть обозначаемого объекта. Например:

```
Summa, Time, i, j, Radius, init
```

Некоторым идентификаторам заранее предписан определенный смысл и их называют стандартными, например `Sin` — это имя известной математической функции.

Описания объектов данных связаны с правилами обработки данных. Данные бывают разные и необходимо для каждого из них определить его свойства. Например, если в качестве данных выступает массив, следует задать его размерность, границы индексов, тип элементов массива. Описательная часть языка программирования является необходимой как для системных программистов — разработчиков трансляторов, которые должны, в частности, проводить синтаксическую и семантическую диагностику программ, — так и для "прикладного" программиста, которому объявления объектов данных облегчают процесс разработки и отладки программ.

В некоторых языках широко применяются описания данных по умолчанию для стандартных числовых и символьных данных. В этом случае тип данных задается с помощью имени объекта данных. Например, в Фортране переменные, имена которых начинаются с букв I, J, K, L, M, N, по умолчанию принимают целые значения (при отсутствии явного описания типа, которое также возможно), т. е. определены как числовые данные целого типа. В Бейсике данные строкового типа присваиваются переменным, имена которых заканчиваются специальным символом $: A$, S1$, а просто имена без $ и специального описания получают действительный тип значений двойной точности.

Особый интерес в языках программирования представляют описания нестандартных структур данных, таких как запись, файл, объект, список, дерево и т. п.

Приведем список наиболее употребительных обозначений типов данных, используемых в описаниях:

Целый — *Integer*

Вещественный — *Real*

Логический — *Boolean*

Символьный — *Char*
Строковый — *String*
Массив — *Array*
Множество — *Set*
Файл — *File*
Запись — *Record*
Объект — *Object*

Переменные играют важнейшую роль в системах программирования. Понятие "переменная" в языках программирования отличается от общепринятого в математике. Переменная — это программный объект, способный принимать некоторое значение с помощью оператора присваивания. В ходе выполнения программы значения переменной могут неоднократно изменяться. Каждая переменная после ее описания связывается с некоторым значением. Синтаксически переменная представляется своим идентификатором (или именем).

Семантический смысл переменной заключается в хранении некоторого значения, соответствующего ее типу (например, переменная целого типа может принимать значение произвольного целого числа), а также в выполнении с ней операций пересылки в нее и извлечения из нее этого значения.

Функция — программный объект, задающий вычислительную процедуру определения значения, зависимого от некоторых аргументов. Вводится в языки программирования для задания программистом необходимых ему функциональных зависимостей. В каждом языке высокого уровня имеется в наличии библиотека стандартных функций: арифметических, логических, символьных, файловых и т. п. Функции — стандартные и задаваемые программистом — используются в составе выражений.

Выражения строятся из постоянных величин, имен переменных, функций, скобок, знаков операций и т. д. Выражение имеет определенный тип, определяемый типом принимаемых в итоге его вычисления значений. Возможны выражения арифметические, принимающие числовые значения, логические, символьные, строковые и т. д. Выражение 5 + 7 является, несомненно, арифметическим, выражение A + B может иметь самый разный смысл — в зависимости от типа данных, которые стоят за идентификаторами A и B.

Процедура — программный объект, представляющий некоторый самостоятельный этап обработки данных. По сути, процедуры явились преемниками подпрограмм, которые были введены для облегчения разработки программ еще на самых ранних стадиях развития программирования. При своем описании процедура имеет входные и выходные параметры, называемые формаль-

ными. При использовании процедуры ее параметры, получившие конкретные значения, становятся фактическими.

Модуль (Unit) — это специальная программная единица, предназначенная для создания библиотек и разделения больших программ на логически связанные блоки.

По сути, модуль — набор констант, типов данных, переменных, процедур и функций. В состав модуля входят разделы: заголовок, интерфейс, реализация, инициализация.

Заголовок необходим для ссылок на модуль.

Интерфейс содержит объявления, включая процедуры и функции.

Раздел *реализация* содержит тела процедур и функций, перечисленных в интерфейсной части.

Раздел *инициализация* содержит операторы, необходимые для инициализации модуля.

Каждый модуль компилируется отдельно, и каждый элемент модуля можно использовать в программе без дополнительного объявления.

Контрольные вопросы

1. Какие преимущества имеют языки программирования высокого уровня по сравнению с машинно-ориентированными языками?
2. Каковы основные составляющие языка программирования высокого уровня?
3. В чем отличие понятий языков программирования от аналогичных понятий естественного языка?
4. С какой целью используются и что представляют собой металингвистические формулы Бэкуса — Наура?
5. Что представляет собой синтаксическая диаграмма Вирта?
6. В чем различие между постоянными и переменными величинами? Чем характеризуется величина?
7. В чем отличие между величинами простыми и структурированными?
8. Для чего служит описание величин в программах? Какие стандартные типы данных используют языки высокого уровня?

9. Для чего служат операторы?
10. В чем состоит назначение функций? процедур? модулей?

Темы для рефератов и докладов

1. Проблемы изучения естественных и формальных языков.
2. Базовые понятия математической лингвистики.
3. Метаязыки для описания формальных языков.
4. Тенденции развития языков программирования высокого уровня.
5. Развитие описания данных в языках программирования.
6. Развитие декларативных языков программирования.

Вопросы для обсуждения

1. Какими могут быть языки "сверхвысокого" уровня, делающие ненужным программирование?
2. Кому больше нужны БНФ и диаграммы Вирта: программистам или разработчикам компиляторов?

Задачи и упражнения

1. Запишите на основе алфавита, включающего русские и английские буквы, цифры и специальные знаки (по своему усмотрению), несколько цепочек символов. Опишите эти цепочки на метаязыках БНФ и диаграмм Вирта.
2. Опишите известные вам конструкции языков программированиия с помощью БНФ.
3. Опишите известные вам операторы языков программированиия с помощью диаграмм Вирта.

Лабораторные работы

1. Найдите в литературе и интернет-источниках описание синтаксиса языков высокого уровня. С помощью каких метаязыков оно выполнено? Составьте справочную таблицу, которая помогла бы быстро найти нужное описание конструкции языка программирования.

5.3. Паскаль как язык структурно-ориентированного программирования

Учебный материал

Введение в Паскаль

Язык Паскаль был создан Н. Виртом в 1971 г. как язык обучения программированию и записи алгоритмов в статьях и книгах. Паскаль стал первым языком, с которым знакомится большинство будущих программистов в мире.

Он оказался настолько удачным, что до сих пор играет особую роль и в практическом программировании, и в его изучении. В нем реализованы принципы структурного программирования и полного описания данных, повышающие устойчивость программного кода, позволяющие найти и устранить большинство ошибок еще на стадии трансляции программы.

Трансляторы для программ, написанных на Паскале, разработаны для различных компьютеров и в настоящее время имеют множество разновидностей.

Существует много версий языка Паскаль. Различия между ними порой весьма велики. Базовая версия языка, созданная Виртом, в этих версиях значительно расширена и дополнена, для того чтобы дать/получить из языка обучения инструмент профессиональных разработчиков прикладного программного обеспечения.

Тем не менее это версии одного языка, что, в частности, подтверждается их совместимостью "сверху вниз", т. е. любая программа, написанная на "младшей" версии языка, останется работоспособной и при переходе к "старшей" версии. Приведенные далее тексты программ и примеры соответствуют (если нет специальных оговорок) практически всем версиям Паскаля.

Любая программа на Паскале является текстовым файлом с собственным именем и с расширением pas. Рассмотрим в качестве примера текст програм-

мы 1 решения квадратного уравнения. Эта программа имеет вид последовательности символов латинских и русских букв, арабских цифр, знаков операций, скобок, знаков препинания и некоторых дополнительных символов. В ней можно выделить описания данных и операторы, описывающие действия, которые надо выполнить компьютеру над этими данными.

Программа 1

```
program SquareEqRoots;                          {заголовок программы}
var                                             {список переменных}
       a,b,c: real;                             {коэффициенты уравнения}
       d,x1, x2: real;                          {вспомогательные переменные}
begin                                           {начало программы}
       writeln;                                 {пропуск строки на экране}
       writeln('введи a,b,c'); read(a,b,c);{запрос и ввод данных}
       d:=b*b-4*a*c;                                    {дискриминант}
       if d<0 then                                      {если d<0, то}
              write('корней нет')                       {печатать}
       else                                             {иначе}
       begin                                            {начало серии команд}
              x1:=(-b+sqrt(d))/(2*a);
              x2:=(-b-sqrt(d))/(2*a);                   {вычисляем корни}
              write('x1=',x1,' x2=',x2)                 {печать корней}
       end                                              {конец серии}
end.                                            {конец программы}
```

Схематически программа представляется в виде последовательности восьми разделов:

1. Заголовок программы.
2. Описание внешних модулей, процедур и функций.
3. Описание меток.
4. Описание констант.
5. Описание типов переменных.
6. Описание переменных.
7. Описание функций и процедур.
8. Раздел операторов.

Не в каждой программе обязательно присутствуют все восемь разделов, в простейшей программе, например, могут быть только 5-й и 8-й разделы.