

МЕТОДЫ ПРОГРАММИРОВАНИЯ КОМПЬЮТЕРНЫЕ ВЫЧИСЛЕНИЯ

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

ЗАДАНИЯ И УПРАЖНЕНИЯ

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЛАБОРАТОРНЫЕ РАБОТЫ

ТЕМЫ РЕФЕРАТОВ И ВОПРОСЫ ДЛЯ ОБСУЖДЕНИЯ

ПРОФИЛЬНАЯ ШКОЛА

**ИНФОРМАТИКА И
ИНФОРМАЦИОННО-
КОММУНИКАЦИОННЫЕ
ТЕХНОЛОГИИ**



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	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Антонина Панюшева</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Екатерина Капальгина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Людмила Минина</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

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	7
МОДУЛЬ 5. ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ.....	9
5.1. История развития языков программирования и парадигмы программирования	11
Учебный материал	11
Контрольные вопросы	17
Темы для рефератов и докладов	17
Вопросы для обсуждения	18
Задачи и упражнения	18
Лабораторные работы.....	18
5.2. Языки программирования высокого уровня. Метаязыки для описания синтаксических конструкций языка высокого уровня	19
Учебный материал	19
Контрольные вопросы	27
Темы для рефератов и докладов	28
Вопросы для обсуждения	28
Задачи и упражнения	28
Лабораторные работы.....	29
5.3. Паскаль как язык структурно-ориентированного программирования	30
Учебный материал	30
Введение в Паскаль.....	30
Основные конструкции языка Паскаль	37
Структуры данных	42
Процедуры и функции	53
Обработка файлов	58
Динамические информационные структуры	64

Работа с графикой	69
Система программирования на Паскале	77
Контрольные вопросы	78
Темы для рефератов и докладов	79
Вопросы для обсуждения	79
Задачи и упражнения	79
Лабораторные работы	81
5.4. Методы и искусство программирования.....	83
Учебный материал	83
Основные принципы разработки и анализа алгоритмов	92
Методы построения алгоритмов, ориентированные на структуры данных	99
Рекурсивные алгоритмы	103
Алгоритмы поиска и сортировки	105
Контрольные вопросы	120
Темы для рефератов и докладов	121
Вопросы для обсуждения	122
Задачи и упражнения	122
Лабораторные работы	123
5.5. Введение в язык программирования Си	124
Учебный материал	124
Общая характеристика языка и пример программы на Си	124
Элементы Си: алфавит, идентификаторы, литералы, служебные слова	129
Типы данных и операции в языке Си. Выражения	132
Операторы. Управляющие конструкции языка	141
Оператор присваивания	141
Оператор <i>if/else</i>	142
Оператор-переключатель <i>switch</i>	144
Оператор цикла <i>for</i>	146
Оператор цикла <i>while</i>	149
Оператор цикла <i>do/while</i>	150
Оператор <i>break</i>	150
Оператор продолжения <i>continue</i>	151
Оператор безусловного перехода <i>goto</i>	152
Составные операторы и блоки	152
Структура программы на Си. Понятие о функциях	153
Классы памяти	160
Функции ввода/вывода	165
Директивы препроцессора	171
Сравнение языков программирования Си и Паскаль	174
Контрольные вопросы	175
Темы для рефератов и докладов	176
Вопросы для обсуждения	176
Задачи и упражнения	177
Лабораторные работы	184

5.6. Элементы объектного программирования	186
Учебный материал	186
Контрольные вопросы	204
Темы для рефератов и докладов	205
Вопросы для обсуждения	205
Задачи и упражнения	205
Лабораторные работы	206
5.7. Основы логического программирования на языке Пролог	207
Учебный материал	207
Общие сведения	207
Алгоритм выполнения программ на Прологе.....	213
Рекурсия	217
Предикат отсечения и управление логическим выводом в программах	220
Обработка списков	222
Решение логических задач на Прологе	226
Контрольные вопросы	229
Темы для рефератов и докладов	230
Вопросы для обсуждения	230
Задачи и упражнения	230
Лабораторные работы	232
МОДУЛЬ 6. КОМПЬЮТЕРНЫЕ ВЫЧИСЛЕНИЯ	233
6.1. Вычислительные методы	235
Учебный материал	235
Вычисление значений функций. Интерполяция.....	235
Решение нелинейных уравнений с одной переменной.....	245
Решение систем линейных уравнений	257
Численное интегрирование	261
Контрольные вопросы	267
Темы для рефератов и докладов	268
Вопросы для обсуждения	269
Задачи и упражнения	270
Лабораторные работы	274
Задания для самостоятельных и контрольных работ.....	280
6.2. Понятие о компьютерном моделировании.....	281
Учебный материал	281
Моделирование как метод познания	281
Этапы и цели компьютерного моделирования	284
Классификация информационных моделей.....	289
Построение компьютерной модели. Моделирование	295
Информационные модели баз данных.....	297
Информационное моделирование в электронных таблицах.....	300

Контрольные вопросы	303
Темы для рефератов и докладов	304
Вопросы для обсуждения	305
Задачи и упражнения	305
Лабораторные работы	308
Литература	316
Предметный указатель.....	317

МОДУЛЬ 5

Языки и методы программирования

- 5.1. История развития языков программирования и парадигмы программирования**
- 5.2. Языки программирования высокого уровня. Метаязыки для описания синтаксических конструкций языка высокого уровня**
- 5.3. Паскаль как язык структурно-ориентированного программирования**
- 5.4. Методы и искусство программирования**
- 5.5. Введение в язык программирования Си**
- 5.6. Элементы объектного программирования**
- 5.7. Основы логического программирования на языке Пролог**

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 называли "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-1" 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-1". 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 высшее 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 "разногласицу" 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 совершенствоваться 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 непроведенным 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.

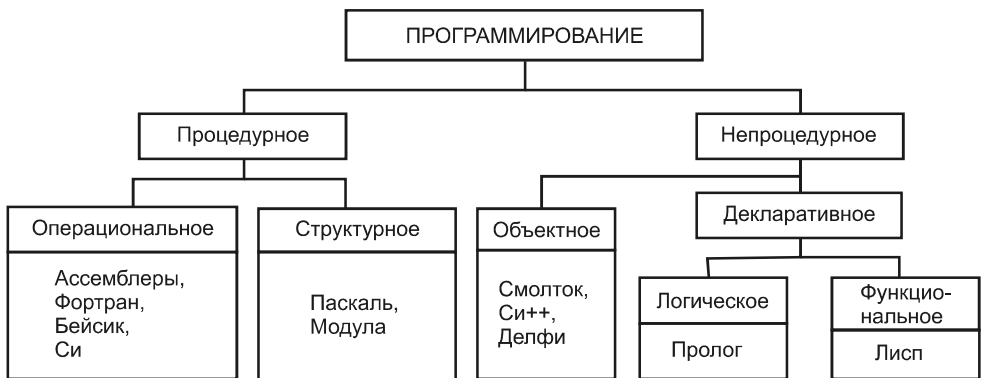


Рис. 5.1. Классификация языков программирования

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> ::= .

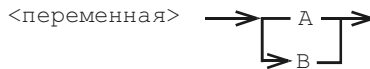
В большинстве учебных пособий по программированию, технических описаний языков, метаформулы рассматриваемого языка представлены полностью.

Синтаксические диаграммы позволяют графически отобразить значения метaperеменных 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.

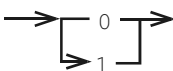
Синтаксическая диаграмма



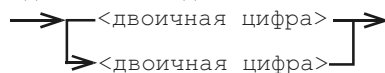
эквивалентна метаформуле <переменная> ::= A|B.

Еще примеры:

<двоичная цифра> ::=



<двоичный код> ::=



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;