## § 6. АЛГОРИТМ И ЕГО СВОЙСТВА

### 6.1. РАЗЛИЧНЫЕ ПОДХОДЫ К ПОНЯТИЮ «АЛГОРИТМ»

Понятие алгоритма - одно из фундаментальных понятий информатики. Алгоритмизация наряду с моделированием выступает в качестве общего метода информатики. К реализации определенных алгоритмов сводятся процессы управления в различных системах, что делает понятие алгоритма близким и кибернетике.

Алгоритмы являются объектом систематического исследования пограничной между математикой и информатикой научной дисциплины, примыкающей к математической логике - **теории алгоритмов.**

Особенность положения состоит в том, что при решении практических задач, предполагающих разработку алгоритмов для реализации на ЭВМ, и тем более при использовании на практике информационных технологий, можно, как правило, не опираться на высокую формализацию данного понятия. Поэтому представляется целесообразным познакомиться с алгоритмами и алгоритмизацией на основе содержательного толкования сущности понятия алгоритма и рассмотрения основных его свойств. При таком подходе алгоритмизация более выступает как набор определенных практических приемов, особых специфических навыков рационального мышления в рамках заданных языковых средств. Можно провести аналогию между этим обстоятельством и рассмотренным выше подходом к измерению информации: тонкие математические построения при «кибернетическом» подходе не очень нужны при использовании гораздо более простого «объемного» подхода при практической работе с компьютером.

Само слово «алгоритм» происходит от algorithmi - латинской формы написания имени великого математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических действий. Первоначально под алгоритмами и понимали только правила выполнения четырех арифметических действий над многозначными числами.

### 6.2. ПОНЯТИЕ ИСПОЛНИТЕЛЯ АЛГОРИТМА

Понятие исполнителя невозможно определить с помощью какой-либо формализации. Исполнителем может быть человек, группа людей, робот, станок, компьютер, язык программирования и т.д. Важнейшим свойством, характеризующим любого из этих исполнителей, является то, что исполнитель умеет выполнять некоторые команды. Так, исполнитель-человек умеет выполнять такие команды как «встать», «сесть», «включить компьютер» и т.д., а исполнитель-язык программирования Бейсик - команды PRINT, END, LIST и другие аналогичные. Вся совокупность команд, которые данный исполнитель умеет выполнять, называется системой команд **исполнителя** (СКИ).

В качестве примера (рис. 1.11) рассмотрим исполнителя-робота, работа которого состоит в собственном перемещении по рабочему полю (квадрату произвольного размера, разделенному на клетки) и перемещении объектов, в начальный момент времени находящихся на «складе» (правая верхняя клетка).



*Рис. 1.11.* Исполнитель-робот

Одно из принципиальных обстоятельств состоит в том, что исполнитель не вникает в смысл того, что он делает, но получает необходимый результат. В таком случае говорят, что исполнитель действует **формально,** т. е. отвлекается от содержания поставленной задачи и только строго выполняет некоторые правила, инструкции.

Это - важная особенность алгоритмов. Наличие алгоритма формализует процесс решения задачи, исключает рассуждение исполнителя. Использование алгоритма дает возможность решать задачу формально, механически исполняя команды алгоритма в указанной последовательности. Целесообразность предусматриваемых алгоритмом действий обеспечивается точным анализом со стороны того, кто составляет этот алгоритм.

Введение в рассмотрение понятия «исполнитель» позволяет определить алгоритм как понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленной цели. В случае исполнителя-робота мы имеем пример алгоритма «в обстановке», характеризующегося отсутствием каких-либо величин. Наиболее же распространенными и привычными являются алгоритмы работы с величинами - числовыми, символьными, логическими и т.д.

### 6.3. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ АЛГОРИТМОВ

Алгоритм, составленный для некоторого исполнителя, можно представить различными способами: с помощью графического или словесного описания, в виде таблицы, последовательностью формул, записанным на алгоритмическом языке (языке программирования). Остановимся на графическом описании алгоритма, называемом блок-схемой. Этот способ имеет ряд преимуществ благодаря наглядности, обеспечивающей, в частности, высокую «читаемость» алгоритма и явное отображение управления в нем.

Прежде всего определим понятие блок-схемы. Блок-схема - это ориентированный граф, указывающий порядок исполнения команд алгоритма; вершины такого графа могут быть одного из трех типов (рис. 1.12).



*Рис. 1.12.* Три типа вершин графа

На рис. 1.12 изображены «функциональная» (*a*) вершина (имеющая один вход и один выход); «предикатная» *(б)* вершина, имеющая один вход и два выхода (в этом случае функция *Р* передает управление по одной из ветвей в зависимости от значения *Р (Т,* т.е. true, означает «истина», *F,* т.е. false - «ложь»); «объединяющая» (в) вершина (вершина «слияния»), обеспечивающая передачу управления от одного из двух входов к выходу. Иногда вместо Т пишут «да» (либо знак +), вместо *F-* «нет» (либо знак -).

Из данных элементарных блок-схем можно построить четыре блок-схемы (рис. 1.13), имеющих особое значение для практики алгоритмизации.



*Рис. 1.13.* Основные алгоритмические структуры

На рис. 1.13 изображены следующие блок-схемы: *а -* **композиция,** или следование; *б -* **альтернатива,** или **развилка,** *в* и ***г -*** блок-схемы, каждую из которых называют **итерацией,** или **циклом** (с предусловием (*в*), с постусловием (*г*)). *S1* и *S2* представляют собой в общем случае некоторые серии команд для соответствующего исполнителя, *В -* это условие, в зависимости от истинности *(Т)* или ложности *(F)* которого управление передаётся по одной из двух ветвей. Можно доказать, что для составления любого алгоритма достаточно представленных выше четырех блок-схем, если пользоваться их последовательностями и/или суперпозициями.

Блок-схема «альтернатива» может иметь и сокращенную форму, в которой отсутствует ветвь *S2* (рис. 1.14, *а).* Развитием блок-схемы типа альтернатива является блок-схема «выбор» (рис. 1.14, б).



*Рис. 1.14.* Развитие структуры типа «альтернатива»;

о) - неполная развилка; *б) -* структура «выбор»

На практике при составлении блок-схем оказывается удобным использовать и другие графические знаки (некоторые из них приведены на рис. 1.15).



*Рис. 1.15.* Некоторые дополнительные конструкции для изображения блок-схем алгоритмов

### 6.4. СВОЙСТВА АЛГОРИТМОВ

Алгоритм должен быть составлен таким образом, чтобы исполнитель, в расчете на которого он создан, мог однозначно и точно следовать командам алгоритма и эффективно получать определенный результат. Это накладывает на записи алгоритмов ряд обязательных требований, суть которых вытекает, вообще говоря, из приведенного выше неформального толкования понятия алгоритма. Сформулируем эти требования в виде перечня свойств, которым должны удовлетворять алгоритмы, адресуемые заданному исполнителю.

1. Одно из первых требований, которое предъявляется к алгоритму, состоит в том, что описываемый процесс должен быть разбит на последовательность отдельных шагов. Возникающая в результате такого разбиения запись представляет собой упорядоченную совокупность четко разделенных друг от друга предписаний (директив, команд, операторов), образующих прерывную (или, как говорят, дискретную) структуру алгоритма. Только выполнив требования одного предписания, можно приступить к выполнению следующего. Дискретная структура алгоритмической записи может. Например, подчеркиваться сквозной нумерацией отдельных команд алгоритма, хотя это требование не является обязательным. Рассмотренное свойство алгоритмов называют дискретностью.

2. Используемые на практике алгоритмы составляются с ориентацией на определенного исполнителя. Чтобы составить для него алгоритм, нужно знать, какие команды этот исполнитель может понять и исполнить, а какие - не может. Мы знаем, что у каждого исполнителя имеется своя система команд. Очевидно, составляя запись алгоритма для определенного исполнителя, можно использовать лишь те команды, которые имеются в его СКИ. Это свойство алгоритмов будем называть понятностью.

3. Будучи понятным, алгоритм не должен содержать предписаний, смысл которых может восприниматься неоднозначно, т.е. одна и та же команда, будучи понятна разным исполнителям, после исполнения каждым из них должна давать одинаковый результат.

Запись алгоритма должна быть настолько четкой, полной и продуманной в деталях, чтобы у исполнителя не могло возникнуть потребности в принятии решений, не предусмотренных составителем алгоритма. Говоря иначе, алгоритм не должен оставлять места для произвола исполнителя. Кроме того, в алгоритмах недопустимы также ситуации, когда после выполнения очередной команды алгоритма исполнителю неясно, какая из команд алгоритма должна выполняться на следующем шаге.

Отмеченное свойства алгоритмов называют **определенностью** или **детерминированностью.**

4. Обязательное требование к алгоритмам - **результативность.** Смысл этого требования состоит в том, что при точном исполнении всех предписаний алгоритма процесс должен прекратиться за конечное число шагов и при этом должен получиться определенный результат. Вывод о том, что решения не существует - тоже результат.

5. Наиболее распространены алгоритмы, обеспечивающие решение не одной конкретной задачи, а некоторого класса задач данного типа. Это свойство алгоритма называют **массовостью.** В простейшем случае массовость обеспечивает возможность использования различных исходных данных.

### 6.5. ПОНЯТИЕ АЛГОРИТМИЧЕСКОГО ЯЗЫКА

Достаточно распространенным способом представления алгоритма является его запись на алгоритмическом **языке,** представляющем в общем случае систему обозначений и правил для единообразной и точной записи алгоритмов и исполнения их. Отметим, что между понятиями «алгоритмический язык» и «языки программирования» есть различие; прежде всего, под исполнителем в алгоритмическом языке может подразумеваться не только компьютер, но и устройство для работы «в обстановке». Программа, записанная на алгоритмическом языке, не обязательно предназначена компьютеру. Практическая же реализация алгоритмического языка -отдельный вопрос в каждом конкретном случае.

Как и каждый язык, алгоритмический язык имеет свой словарь. Основу этого словаря составляют слова, употребляемые для записи команд, входящих в систему команд исполнителя того или иного алгоритма. Такие команды называют простыми командами. В алгоритмическом языке используют слова, смысл и способ употребления которых задан раз и навсегда. Эти слова называют служебными. Использование служебных слов делает запись алгоритма более наглядной, а форму представления различных алгоритмов - единообразной.

Алгоритм, записанный на алгоритмическом языке, должен иметь название. Название желательно выбирать так, чтобы было ясно, решение какой задачи описывает данный алгоритм. Для выделения названия алгоритма перед ним записывают служебное слово АЛГ (АЛГоритм). За названием алгоритма (обычно с новой строки) записывают его команды. Для указания начала и конца алгоритма его команды заключают в пару служебных слов НАЧ (НАЧало) и КОН (КОНец). Команды записывают последовательно.

Последовательность записи алгоритма:

АЛГ название алгоритма

НАЧ

 серия команд алгоритма

КОН

Например, алгоритм, определяющий движение исполнителя-робота, может иметь вид:

АЛГ в\_склад

НАЧ

 вперед

 поворот на 90° направо

 вперед

КОН

При построении новых алгоритмов могут использоваться алгоритмы, составленные ранее. Алгоритмы, целиком используемые в составе других алгоритмов, называют вспомогательными алгоритмами. Вспомогательным может оказаться любой алгоритм из числа ранее составленных. Не исключается также, что вспомогательным в определенной ситуации может оказаться алгоритм, сам содержащий ссылку на вспомогательные алгоритмы.

Очень часто при составлении алгоритмов возникает необходимость использования в качестве вспомогательного одного и того же алгоритма, который к тому же может быть весьма сложным и громоздким. Было бы нерационально, начиная работу, каждый раз заново составлять и запоминать такой алгоритм для его последующего использования. Поэтому в практике широко используют, так называемые, встроенные (или стандартные) вспомогательные алгоритмы, т.е. такие алгоритмы, которые постоянно имеются в распоряжении исполнителя. Обращение к таким алгоритмам осуществляется так же, как и к «обычным» вспомогательным алгоритмам. У исполнителя-робота встроенным вспомогательным алгоритмом может быть перемещение в склад из любой точки рабочего поля; у исполнителя-язык программирования Бейсик это, например, встроенный алгоритм «SIN».

Алгоритм может содержать обращение к самому себе как вспомогательному и в этом случае его называют **рекурсивным.** Если команда обращения алгоритма к самому себе находится в самом алгоритме, то такую рекурсию называют прямой. Возможны случаи, когда рекурсивный вызов данного алгоритма происходит из вспомогательного алгоритма, к которому в данном алгоритме имеется обращение. Такая рекурсия называется **косвенной.** Пример прямой рекурсии:

АЛГ движение

НАЧ

 вперед

 вперед

 вправо

 движение

КОН

Алгоритмы, при исполнении которых порядок следования команд определяется в зависимости от результатов проверки некоторых условий, называют **разветвляющимися.** Для их описания в алгоритмическом языке используют специальную составную команду - команда **ветвления.** Она соответствует блок-схеме «альтернатива» и также может иметь полную или сокращенную форму. Применительно к исполнителю-роботу условием может быть проверка нахождения робота у края рабочего поля (край/не\_край); проверка наличия объекта в текущей клетке (есть/нет) и некоторые другие:

ЕСЛИ условие ЕСЛИ условие ЕСЛИ край

 ТО серия 1 ТО серия ТО вправо

 ИНАЧЕ серия2 ВСЕ ИНАЧЕ вперед

ВСЕ ВСЕ

Ниже приводится запись на алгоритмическом языке команды выбора (см. рис. 1.14, б), являющейся развитием команды ветвления:

ВЫБОР

 ПРИ условие 1: серия 1

 ПРИ условие 2: серия 2

 …

 ПРИ условие N: серия N

 ИНАЧЕ серия N+1

ВСЕ

Алгоритмы, при исполнении которых отдельные команды или серии команд выполняются неоднократно, называют циклическими. Для организации циклических алгоритмов в алгоритмическом языке используют специальную составную команду цикла. Она соответствует блок-схемам типа «итерация» и может принимать следующий вид:

ПОКА условие НЦ

 НЦ серия

 Серия ДО условие

 КЦ КЦ

В случае составления алгоритмов работы с величинами можно рассмотреть и другие возможные алгоритмические конструкции, например, цикл с параметром или выбор. Подробно эти конструкции будут рассматриваться при знакомстве с реальными языками программирования.

В заключение данного параграфа приведем алгоритм, составленный для исполнителя-робота, по которому робот переносит все объекты со склада в левый нижний угол рабочего поля (поле может иметь произвольные размеры):

АЛГ перенос АЛГ в\_угол3 АЛГ до\_края

НАЧ НАЧ НАЧ

 в\_угол3 до\_края ПОКА не\_край

 ЕСЛИ есть вправо НЦ

 ТО до\_края вперед

 взять вправо КЦ

 в\_угол3 КОН КОН

 установить

 перенос

 ИНАЧЕ

 в\_угол3

 ВСЕ

КОН