**Алгоритмы и программы**

1.Понятие алгоритма.

2.Характерные свойства алгоритмов.

3.ЭВМ как универсальный Исполнитель.

Внешние устройства ЭВМ.

Центральные устройства ЭВМ.

Понятие о машинном языке.

**1.Понятие алгоритма.**

Точное математическое определение алгоритма и изучение этого понятия - предмет спе­циальной математической дисциплины - теории алгоритмов, которая, в свою очередь, опирается на аппарат математической логики. Здесь мы рассмотрим на содержательном (неформальном) уров­не лишь характерные основные черты этого понятия.

Алгоритм - это некоторое правило преобразования информации, применение которого к заданной (исходной) информации приводит к резуль­тату - новой информации.

Так, например, применение правила сложения дробей к 1/2 и 2/3 приводит к результату 5/6.

Основное внимание в теории алгоритмов уделяется методам задания (описания, констру­ирования) алгоритмов.

Алгоритм - это конечный набор инструкций по преобразованию информации (команд), выполнение которых приводит к результату. Каждая инструкция алгоритма содержит точное описание некоторого эле­ментарного действия по преобразованию информации, а также (в явном или неявном виде) указание на инструкцию, которую необходимо выполнить следующей.

Так, алгоритм сложения дробей можно задать следующей последовательностью команд:

Пример 1. Алгоритм сложения дробей.

Вход: A/B, C/D;

1. Вычислить Y = B\*D; {Перейти к следующей команде}

2. Вычислить X1 = A\*D; {Перейти к следующей команде}

3. Вычислить X2 = B\*C; {Перейти к следующей команде}

4. Вычислить X = X1+X2; {Перейти к следующей команде}

5. Вычислить Z = НОД(X,Y); {Перейти к следующей команде}

6. Вычислить Е = X div Z; {Перейти к следующей команде}

7. Вычислить F = Y div Z; {Закончить работу}.

Выход: E/F

Наш алгоритм состоит из 7-ми инструкций, каждая из которых содержит описание одного из арифметических действий над целыми числами: сложение, умножение, вычисление НОД и целочисленное деление div. Кроме того, каждая инструкция (в неявном виде) содержит указание на следующую выполняемую инструкцию. Таким образом, алгоритм описывает детали­зиро­ванный по шагам процесс преобразования информации. Уровень детализации описания опреде­ляется набором допустимых команд. Этот набор называют набором команд Исполнителяили Интерпретатора. При этом подразумевается, что алгоритмы выполняет Исполнитель (Интерпретатор**)** - некоторое устройство, однозначно распозна­ющее и точно выполняющее (интерпретирующее) каждую команду алгоритма.

Для выполнения нашего алгоритмаИсполнитель должен, очевидно, уметь оперировать с целыми числами, выделять числители и знаменатели дробей, а также составлять из пары целых чисел дробь. Кроме того, Исполнитель должен уметь запоминать результаты выполнения опера­ций и переходить к выполнению следующей команды.

Представим себе, что в нашем распоряжении находится Исполнитель, интерпретирующий команды - операции целочисленной арифметики - сложение, вычитание, умножение, вычисление неполного частного (div) и остатка (mod), вычисление НОД и НОК с запоминанием результатов и умением переходить к следующей команде. Тогда для этого Исполнителя можно составлять самые разнообразнные алгоритмы арифметических вычислений - т.е. вычислений, заданных формулами типа X = НОД((A + B) div 100, (A \* B - 7) mod 10) используя команды, аналогичные командам алгоритма из примера 1.

Пример 2. Алгоритм деления отрезка пополам с помощью циркуля и линейки.

Вход: Отрезок AB;

Построить окружность O1 с центром A и радиусом AB;

Построить окружность O2 с центром B и радиусом AB;

Найти точки С и D пересечения окружностей O1 и O2;

Построить отрезок CD;

Найти точку Е пересечения AB и CD.

Выход: Точка Е - середина отрезка AB.

В примере 2 Исполнительобладает набором команд, с помощью которых можно решать геометрические задачи на построения с помощью циркуля и линейки.

Исполнение алгоритма заключается в последовательном выполнении каждого построения и переходе к исполнению следующей команды.

Отметим. что нумеровать команды алгоритма не нужно, если Исполнительвсегда перехо­дит к следующей команде.

Пример 3. Алгоритм решения приведенного квадратного уравнения x2 + px + q = 0;

Вход: Коэффициенты p и q уравнения x2 + px + q = 0;

Вычислить D = p2 - 4q;

Если D < 0 то (ответить “Решений нет”; Перейти к 1);

Если D = 0 то (вычислить x = -p/2; Перейти к 1);

Вычислить x1 = (-p+√(D))/2;

Вычислить x2 = (-p-√(D))/2);

1:Закончить работу.

Выход“Решений нет” или корень x или корни x1, x2.

В этом примере используется команда вида

Если <условие>

то (<последовательность команд>)

Выполняя эту команду, Исполнитель проверяет истинность условия. Если условие выполнено, Исполнитель переходит к выполнению первой команды из последовательности команд, стоящей после слова то и команды, и исполнение алгоритма следует по тому или иному пути в заключенной в скобки. Если же условие не выполнено, Исполнитель переходит к выполнению следующей команды. Такие команды называют выбирающими, условными или ветвлениями. Выбирающие команды содержат в себе другие команды, выполняющиеся в зависимости от результатов проверки условий.

Второй характерной командой, используемой в примере, является команда перехода. Она имеет вид Перейти к < N >, причем число N используется в записи алгоритма как специальная отметка некоторой команды. В примере используются команды перехода Перейти к 1, а числом 1 отмечена команда 1:Закончить работу.

Выполнение команды перехода заключается в том, что Исполнитель переходит к выполне­нию команды, отмеченной отметкой N (нарушая при этом естественную после­дова­тельность выполнениия команд).

**2. Характерные свойства алгоритмов.**

Понятию алгоритма присущи следующие свойства:

1. Элементарность. Каждая команда из набора команд Исполнителясодержит указание выполнить некоторое элементарное (не детализируемое более подробно) действие, однозначно понимаемое и точно выполняемое Исполнителем.

Понятие элементарности поэтому относительно. Так, в алгоритме примера 1 содержится команда вычисления НОД двух чисел. Это означает, что Исполнитель умеет находить НОД, причем алгоритм вычисления (алгоритм Евклида или какой нибудь другой) скрыт от человека, составляющего алгоритмы для этого Исполнителя. Если набор команд Исполнителя не содержит команды вычисления НОД, вычисление НОД должно быть определено в виде алгоритма.

Пример 4**.** Алгоритм Евклида вычисления наибольшего общего делителя целых положительных чисел A и B: НОД(A, B).

Вход A, B;

Вычислить U = A;

Вычислить V = B;

Пока U <> V выполнять

Если U < V

то Вычислить V = V - U

иначе Вычислить U = U - V;

Вычислить D = U.

Выход: D - наибольший общий делитель A и B.

В этом примере использована команда повторения.Она имеет вид

Пока <Условие> выполнять <Команда>

Выполняя эту команду, Исполнитель проверяет истинность Условия. Если Усло­вие истин­но, Исполнитель выполняет Команду, указанную после слова выпол­нять и повторяет проверку Условия. Выполнение Команды и проверка Условия повторяются до тех пор, пока Условие истин­­но. Если Условие ложно, Исполнитель переходит к выполнению команды, следующей за командой пов­то­рения. В этом же примере использована еще одна разновидность команды ветвления - команда вида

Если<Условие> то <Команда 1> иначе < Команда 2>

Выполняя эту команду, Исполнитель проверяет Условие. Если Условие выполнено, выпол­няется Команда 1, в противном случае выполняется команда 2. Далее Исполнитель переходит к сле­дующей команде. Заметим, что команда повторения, как и команды ветвления, содержат в себе другие команды.

2.Определенность. Исполнение алгоритма строго определено. Это означает, что на каждом шаге Исполнительне только точно выполняет команду, но и однозначно определяет следующую исполняемую команду. Поэтому повторное выполнение алгоритма для одних и тех же исходных данных в точности повторяет первое его выполнение.

Так, в исполнении алгоритма в примере 3 возможны ветвления, которые, однако, одно­значно определены условиями D < 0, D = 0.

3.Массовость. Алгоритмы, как правило, описывают ход решения не одной-единственной задачи, а целого класса однотипных задач.

Так, в примере 2 описан алгоритм сложения любых двух дробей. Одна-единственная задача, решаемая Исполнителем в данный момент, определена значениями исходных данных A, B, C, D. Изменение исходных данных означает решение другой задачи из этого же класса задач.

Аналогично, алгоритм примера 2 строит середину любого отрезка, заданного его концами, а в примере 3 с помощью алгоритма решается любое приведенное квадратное уравнение.

4. Результативность. Исполение любого алгоритма должно быть закончено через конеч­ное число шагов (т.е. выполнение конечного числа команд) с некоторым результатом.

Так, в примере 2 результат - точка E - середина отрезка AB. Алгоритм примера 3 выдает результат “Решений нет “ даже в том случае, когда уравнение не имеет действительных корней, поскольку его дискриминант меньше нуля.

Заметим, однако, что количество шагов алгоритма, решающего некоторую задачу, заранее неизвестно и может быть очень большим, поэтому свойство результативности конкретного алгоритма часто приходится специально доказывать. Так, проверка свойства результативности в примере 4 требует доказательства того факта, что выполнение команды повторения закончится за конечное число шагов.

Рассмотренные примеры показывают, что понятие Исполнителя является одной из основ­ных абстракций, используемых для изучения алгоритмов. Именно система командИсполнителя и есть уточнение набора средств, с помощью которых строятся алгоритмы. В наших примерах системы команд Исполнителяпредметно-ориентированы. Так, в примере 1 Исполнитель имеет дело с цело­численной арифметикой, а в примере 3 - с арифметикой действительных чисел. Особенно наглядно это демонстрирует Исполнительпримера 2, команды которого выполняют геометрические построения. Вместе с тем очевидно, что существуют команды, которые должны входить в систему команд любого Исполнителя, претендующего на универсальность в некоторой предметной области. Это - коман­ды ветвлений, повторения, перехода. Эти команды непосред­ственно не указывают на пребразования данных, а лишь на управление этими преобразованиями.

Несмотря на все многообразие форм представления информации и операций ее преобра­зования, используемых человеком в своей деятельности, оказалось возможным создание универсального Исполнителя, т.е. Исполнителя, система команд которого позволяет промо­де­лировать любую другую систему команд. Таким Исполнителем является ЭВМ.

**3. ЭВМ как универсальный Исполнитель.**

На рис 1 представлена блок-схема **ЭВМ**.



**Внешние устройства ЭВМ.**

Устройства ввода информациипредназначены для ввода информации в ЭВМ. Устройства ввода преобразуют информацию из формы, предназначенной для пользователя, в форму, предназначенную для хранения и обработки вЭВМ - в двоичный код. Наиболее распростра­ненные устройства ввода - клавиатура, сканер, “мышь”, и другие.

Устройства вывода информации предназначены для вывода информации (результатов) в форме, предназначенной для пользователя - в виде чисел, текстов, рисунков и т.д.

Характерными для персональной ЭВМ устройствами вывода являются монитор (дисплей), принтер, плоттер.

Внешние запоминающие устройства (ВЗУ) предназначены для длительного (при выключенной ЭВМ) хранения информации. В настоящее время наиболее распространены накопители на гибких магнитных дисках (НГМД), накопители на жестких магнитных дисках (НМД), накопители на магнитных лентах (НМЛ).

Согласования сигналов, которыми обмениваются устройства ЭВМ в процес­се работы, осу­ществляют интерфейсные блоки (контроллеры). Интерфейсный блок - это совокупность аппарат­ных и программных средств, поддерживающих про­цесс обмена данными между устройствами ЭВМ, различными по быстро­дей­ствию, уровню сигналов, способу кодировки информации.

**Центральные устройства ЭВМ.**

Ядро **ЭВМ** составляют так называемые центральные устройства - центральный процессор и оперативное запоминающее устройство (ЦП и ОЗУ).

Центральные устройства предназначены для оперативного хранения и преобразования информации. Совокупность центральных устройств объединена в системный блок.

Вся информация, необходимая для исполнения алгоритма - исходные, промежуточные и выходные данные, а также сам алгоритм, хранится в ОЗУ в закодированном виде.

ОЗУпредставляет из себя совокупность специальных ячеек, каждая из которых предназначена для хранения двоичного кода информации фиксированного объема. Каждая ячейка памяти имеет свой номер, называемый адресом. В современных ПЭВМ ячейка ОЗУ хранит 1 байт информации - 8 двоичных цифр, называемых битами.

Байт 0 1 2 3 4 5 6 7

[адрес]

Характерной особенностью ОЗУ является то, что доступ к данным, хранящимся там, осуществляется по их адресам. Время доступа к данному не зависит от адреса ячейки, в которой оно хранится. Поэтому ОЗУ называют памятью с прямым (случайным) доступом. Размером (объемом) ОЗУназывают количество ее ячеек. Размер ОЗУ выражают в байтах, килобайтах (Kb) и мегабайтах (Mb). 1 килобайт = 1024 байта, 1 мегабайт = 1024 килобайта. И данные, и команды как правило, занимают в ОЗУ несколько подряд адресованных байтов.

Центральный процессор - устройство, предназначенное для исполнения алгоритммов, хранимых в ОЗУ в виде набора команд. Каждый центральный процессор имеет свою систему командИсполнителя. Система команд реального процессора содержит десятки команд, и ее изучение - предмет отдельного курса обучения. Мы рассмотрим лишь основные принципы построения машинного языка.

**Понятие о машинном языке.**

Набор команд процессора содержит:

* арифметико-логические команды - команды арифметических действий над двоичными числами и логических действий над двоичными векторами;
* команды управления - команды перехода, ветвлений, повторений, и некоторые другие команды;
* команды пересылки данных - команды, с помощью которых обмениваются данными ОЗУ иЦП;
* команды ввода-вывода данных - команды, с помощью которых обмениваются данными ЦПи внешние устройства.

Каждая команда содержит код операции, ею выполняемой и информацию об адресах данных, над которыми эта операция выполняется. Кроме этого, команда (непосредственно - команды управления и косвенно - другие команды) содержит информацию об адресе команды, которая будет выполняться следующей. Таким образом, любая последовательность команд, размещенная в ОЗУ, фактически представляет из себя алгоритм, записанный в системе команд процессора - машинную программу.

Наиболее распространенной сейчас является схема ЭВМ с общей шиной. Общая шина - это центральная информационная магистраль, связывающая внешние устройства с центральным процессором. Она состоит из шины данных, шины адреса и шины управления. Шина данных предназначена для обмена данными междуОЗУ и внешними устройствами. По шине адреса передаются адреса данных. Эта шина одно направление. Шина управления служит каналом обмена управляющими сигналами между внешними устройствами и центральным процессором.

Таким образом, язык процессора — это набор команд, каждая из которых описывает неко­торое элементарное действие по преобразованию информации, представленной в двоичном коде. Универсальное использование двоичного кода представления информации самых разно­об­раз­ных форм приводит к тому, что программа решения даже достаточно простой задачи содержит сотни машинных команд. Написать такую программу, используя машинные команды, весьма непросто даже квалифицированному программисту. Реальные программы состоят из десятков и сотен тысяч машинных команд. Поэтому любая технология проектирования программы должна опираться на приемы, характерные для человеческого мышления, оперировать привычными для человека понятиями из той предметной области, которой принадлежит задача.

Иными словами, программист (проектировщик алгоритмов) должен иметь возможность сформулировать свой алгоритм на языке привычных понятий; затем специальная программа должна выразить эти понятия с помощью машинных средств, осуществляя перевод (трансляцию) текста алгоритма на язык машины.

Эта необходимость и привела к появлению языков программирования высокого уровня как языков записи алгоритмов, предназначенных для исполнения на ЭВМ**.**