МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО
ОБРАЗОВАНИЯ УССР

ОДЕССКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им.И.И.МЕЧНИКОВА

В.С.Макогон

ОСНОВЫ АЛГОРИТМИЗАЦИИ В ИНФОРМАТИКЕ

ОДЕССА 1986

МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО
ОБРАЗОВАНИЯ УССР

ОДЕССКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им.И.И.МЕЧНИКОВА

В.С.Макогон

ОСНОВЫ АЛГОРИТМИЗАЦИИ В ИНФОРМАТИКЕ

Утверждено

Ученым Советом университета в качестве
учебного пособия

ОДЕССА ОГУ 1986

УДК 519.6

Основы алгоритмизации в информатике : Учеб.пособие / В.С.Макогон.- Одесса : ОГУ, 1986. - 72 с

Учебное пособие подготовлено в соответствии с программой по дисциплине "Научно-методические основы информатики и вычислительной техники" для педагогических специальностей вузов.

В пособии освещены: первоначальные сведения об ЭВМ, этапы решения задач с применением компьютеров, свойства алгоритмов, способы их описания; рассмотрены основы конструирования, записи алгоритмов на учебном алгоритмическом языке.

Рассчитано на студентов, изучающих основы информатики и вычислительной техники, а также может быть использовано при обучении основам алгоритмизации и программирования в средней школе.

список лит. - 4 назв.

Рецензенты : кандидат физико-математических наук Т.С.Зверкова, методист Одесского областного институту усовершенствования учителей В.В.Гринчук

Ответственный редактор: кандидат физико-математических наук Н.Я.Тихоненко

Одесский Ордена Трудового Красного Знамени государственный университет им.И.И.Мечникова

В В Е Д Е Н И Е

В начале 1985 года Постановлением ЦК КПСС и Совета Министров была рассмотрена и одобрена Общегосударственная программа создания, развития производства и эффективного использования вычислительной техники и автоматизированных систем на период до 2000 года. Намечены пути перевооружения народного хозяйства СССР на основе вычислительной техники и микроэлектроники в интересах резкого повышения производительности труда в материальном производстве, существенного совершенствования управления на всех уровнях руководства и принятия решений.

Выполнение намеченных задач возможно, однако, лишь в том случае, если "понимание роли вычислительной техники в развитии общества, способность применить ее в своем деле, знание основ и н ф о р м а т и к и - науки о решении задач с помощью ЭВМ станут своего рода в т о р о й г р а м о т н о с т ь ю каждого образованного человека." [3]

Вот почему одним из главных положений школьной реформы стала задача введения информатики и электронно-вычислительной техники в учебно-воспитательный процесс школы и обеспечения компьютерной грамотности учащейся молодежи. В всех средних учебных заведениях страны введен курс "Основы информатики и вычислительной техники", предусматривается проведение широкого эксперимента по использованию ЭВМ в преподавании школьных предметов.

Создание ЭВМ можно сравнить с самыми выдающимися открытиями человечества - такими, как добывание огня, изобретение колеса, освоение выплавки металлов, создание паровой машины, освоение электричества, использование атомной энергии. Однако в этом ряду ЭВМ занимает особое место: если обычные машины расширяли физические возможности человека в процессе производства, то вычислительные машины расширили его интеллектуальные возможности. Они являются одним из важнейших факторов превращения науки в непосредственную производительную силу нашего общества. Без ЭВМ были бы невозможны космические исследования, атомная энергетика, эффективное развитие науки, экономики и техники в целом.

Благодаря ЭВМ идет интенсивное проникновение методов прикладной математики в другие науки. Важное значение приобрело применение математических методов в экономике; математическое моделирование стало широко применяться в химии геологии, медицине, психологии, лингвистике.

Отметим три основные причины бурного развития производства и широкого распространения ЭВМ в последние годы.

Мы часто пользуемся термином "ЭВМ", подразумевая по привычке только одну способность этих машин – умение быстро вычислять. Так, первая в СССР электронная вычислительная машина – МЭСМ, разработанная под руководством академика С.А.Лебедева и вступившая в строй в 1951 году, могла выполнять арифметические действия над 5 – 6 - значными числами со скоростью около 50 операций в секунду. Заметим, что человек с помощью карандаша и бумаги выполнит одну такую операцию примерно за 30 с. Получается, что наша первая ЭВМ производила вычисления приблизительно в полторы тысячи раз быстрее, чем человек, считавший вручную. Самые мощные современные ЭВМ работают со скоростями в десятки и даже сотни миллионов операций в секунду ! Такого скачка, такого повышения производительности труда не знала ни одна сфера человеческой деятельности за всю историю существования нашей цивилизации.

Однако умение быстро и точно выполнять арифметические расчеты – отнюдь не единственное и даже не главное умение современных вычислительных машин. В последние годы основной объем работы, выполняемой с помощью вычислительной техники, переместился с вычислительных на и н ф о р м а ц и о н н ы е операции. Именно вычислительная техника оказалась наиболее удобной для таких работ, как сбор информации, ее преобразование в различные формы, запись и длительное хранение, быстрая выдача информации по запросам. Эта важнейшая способность вычислительных машин – одна из основных причин интенсивного развития их производства и применения в самых разнообразных сферах человеческой деятельности.

Попутно отметим, что между понятиями "ЭВМ" и "компьютер" нет принципиальной разницы. ЭВМ – это русский аналог английского слова "компьютер". С недавних пор "компьютер" употребляется чаще, вероятно, потому, что в словах "вычислительная машина" упор как бы делается на вычисления, а современный компьютер – много функциональное устройство, которое не только вычисляет, но, как мы

уже говорили, хранит информацию, оперирует числами, словами, зрительными образами, изображаемыми на экране; выдает результаты на печать в виде таблиц, графиков, схем и т.д.

Вторая причина – в появлении достаточно простых и понятных для современного человека форм общения с ЭВМ. Пульт с клавиатурой, похожей на клавиатуру пишущей машинки, позволяет набрать и послать в ЭВМ текст вопроса или поручения, числа и сведения о действиях, которые необходимо с ними произвести, знаки, соответствующие различным арифметическим или логическим операциям и т. д. Обычный телевизор может быть использован для отображения информации, поступающей из ЭВМ. При необходимости этот комплект может быть дополнен печатающим устройством или устройством для записи информации на магнитную ленту бытового магнитофона. Появилась возможность передачи и обмена информацией между ЭВМ по каналам связи. Ведутся работы по вводу информации в ЭВМ обычным человеческим голосом, по "обучению" машин читать таблицы, графики, обрабатывать различные сигналы. Основные составные части ЭВМ – процессор и память.

Обработкой информации в ЭВМ занимается п р о ц е с с о р – "мозг и сердце" любой ЭВМ. Процессор управляет работой всей машины: он записывает информацию в память и читает ее из памяти, принимает запросы пользователя и выдает на т е р м и н а л ы (под терминалом понимают любое устройство связи человека с ЭВМ) различные сообщения, и он же производит вычисления, преобразования и пересылку информации. Прежде процессор состоял из множества связанных между собой блоков различного функционального назначения, производящих арифметические, логические и другие операции и имел довольно большие размеры. Скажем, первая наша ЭВМ МЭСМ, о которой уже упоминалось выше, занимала площадь 50 $м^2$, потребляла 25 квт электроэнергии, а радиоламп в ней было больше шести тысяч и для ее нормальной работы требовались специальные холодильные установки.

Третьей основной причиной нового скачка в развитии вычислительной техники послужило освоение промышленностью микроэлектронной технологии, позволившей одновременно решить обычно несовместимые задачи: с одной стороны, резко увеличить скорость обработки информации и увеличить объем памяти, с другой – стать же резко уменьшить размеры ЭВМ, их энергопотребление и стоимость.

В результате цена ЭВМ становится сравнимой с ценой таких обиходных предметов, как радиоприемник или фотоаппарат. Главная отличительная черта современных компьютеров в том, что с ними в нашу жизнь вошел новый класс средств вычислительной техники – м и к р о п р о ц е с с о р ы. В их основе лежат крошечные (как одна клеточка из тетради "по арифметике") полупроводниковые кристаллы (чаще всего кремния), на поверхности которых в результате очень сложного многоэтапного процесса создается многослойная электронная схема, содержащая все необходимые транзисторы, диоды, конденсаторы, сопротивления и соединения между ними. Специалисты называют их большими интегральными схемами (БИС). Это название они получили не по размерам, а насыщенности элементов : одна такая БИС заменяет сотни, а то и тысячи радиоламп, применявшихся в первых ЭВМ. БИС можно скомпоновать и "обучить" так, что они будут выполнять различные операции: арифметические и логические, операции по запоминанию, преобразованию, пересылке текущей информации. Появилась возможность выполнить в виде одной БИС целый процессор. Такие процессоры и получили название микропроцессоров, а созданные на их основе ЭВМ – микрокомпьютеров. Остальные элементы микрокомпьютеров (память, устройства сопряжения с терминалами) также реализуются в виде БИС. Имеются даже о д н о к р и с т а л ь н ы е микрокомпьютеры, выполненные целиком в виде одной БИС.

Появление микропроцессоров открыло перед вычислительной техникой совершенно неожиданные перспективы. Их можно встраивать в системы автоматического управления; можно вмонтировать прямо в станок, в научный прибор и даже ... в кухонную плиту и другие домашние электроприборы. Взять хотя бы телефон, которым мы пользуемся ежедневно. Достаточно вмонтировать в него микропроцессор – и телефонный аппарат станет надежным секретарем. Он сможет запомнить номера звонивших абонентов, их сообщения; автоматически повторять вызов при сигнале "занято". Существуют и стиральные машины, в которые нужно только загрузить белье и засыпать стиральный порошок – все остальное машина сделает сама : нальет и нагреет воду до требуемой температуры, замочит белье, выстирает, прополощет, отожмет и сольет воду. Есть электроплиты, способные по заданной программе включить духовку, нагреть ее до указанной температуры и подать звуковой сигнал; в процессе выпечки они способны через определенное время снижать температуру и вовсе выключить

плиту при достижении готовности выпекаемого изделия. А как не упомянуть о компьютерных мастерах "на все руки" – роботах, умеющих уже не только подавать, транспортировать, отбраковывать и складировать детали на производстве, но и готовить обед, натирать полы, лазать по деревьям, заниматься электросваркой в труднодоступных местах и сложных условиях.

Все больше появляется микропроцессорных игрушек: их подключают к обычному телевизору и на экране появляется, например, мишень, на которой можно потренироваться в спортивной стрельбе из светового пистолета, или полоса с препятствиями, через которую нужно провести автомобиль. Существуют игры, полностью моделирующие действия шофера, летчика и т.д., вырабатывающие определенные полезные навыки, и превращение компьютера в тренажер способствует, таким образом, развитию творческих, мыслительных, профессиональных способностей.

Наиболее яркими представителями новой техники являются п е р с о н а л ь н ы е электронно-вычислительные машины (ПЭВМ) или персональные к о м п ь ю т е р ы (ПК). Они ориентированы на индивидуальное пользование, подобно магнитофону, телевизору, пишущей машинке. Это инструмент для решения широкого круга задач – от сложных профессиональных до самых разнообразных бытовых. Например, ученый -экспериментатор получит в свое пользование компьютер, который будет хранить все интересующие его справочные данные, автоматизирует проведение эксперимента. Конструктору – проектировщику ПЭВМ поможет определить оптимальную конструкцию: он сможет совершенствовать ее на экране д и с п л е я (терминала с телевизионным экраном), быстро внося все необходимые поправки, а затем без промедления получить на графопостроителе все чертежи и другую конструкторскую документацию.

С успехом будут применять персональные компьютеры в своей работе писатели, композиторы, художники. С помощью ПК можно постепенно накапливать различную текстовую и графическую информацию, а затем в любой момент получить на экране интересующий пользователя ПК текст из архива, изменять и править его по желанию и в любой момент сделать отпечаток текста на бумаге.

Достоинства персональных ЭВМ бесспорны, а их возможности и сфера применения почти безграничны. Они компактны и просты, причем, тенденция к уменьшению их габаритов и снижению стоимости

будет устойчивой. Портативные ПК поместятся, скажем, в портфеле-"дипломате", а их вес не превышает четырех килограммов. В перспективе ПК будут иметь средства ввода зрительной и речевой информации.

Трудно переоценить те возможности, которые открывает применение ПЭВМ в школе. ЭВМ позволит учителю вести более углубленное преподавание и естественнонаучных и гуманитарных предметов. Сегодня в научных лабораториях разрабатываются пакеты прикладных программ для автоматизированного обучения и контроля за успеваемостью учащихся. Порции теоретического материала при работе с таким пакетом будут закрепляться практической работой школьника за дисплеем ЭВМ. При этом компьютер будет настраиваться на индивидуальные особенности ученика. Если тот сделал ошибку при контроле знаний, ЭВМ поправит его и предложит выполнить новое упражнение для закрепления материала. На уроках математики, в частности, персональная ЭВМ сделает наглядными, позволит воплотить в числах многие отвлеченные теоретические построения - продемонстрировать сходимость последовательности к пределу, численно проверить возникающие у учащихся предположения о свойствах геометрических фигур и т.п. Раннее общение с компьютерами поможет сформировать у учащихся особый стиль мышления - а л г о р и т м и ч е с к и й . Он предполагает умение продумывать способы достижения поставленной цели, расчленять их на элементарные действия и планировать требуемые для их выполнения ресурсы. Чтобы микропроцессорная техника служила обществу с полной отдачей, понадобится м а с с о в а я подготовка людей к пользованию его. И начинать эту подготовку надо со школьной скамьи!

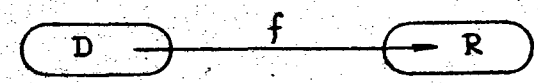
Где бы ни работал, какую бы профессию ни выбрал сегодняшний выпускник школы, он рано или поздно встретится с вычислительной машиной. И тогда многое будет зависеть от того, насколько наши выпускники школ готовы к такой встрече. По словам академика А.П.Ершова, современный человек, не умеющий работать с ЭВМ, вскоре может оказаться в положении неграмотного в библиотеке. Развитие микропроцессорных устройств, создание гибких автоматизированных производств (ГАП), роботов - требуют рабочих, владеющих основами электронной вычислительной техники и программного управления. Ученые стараются упростить диалог с вычислительными машинами до такой степени, чтобы ими могли пользоваться люди без большой специальной подготовки. Вот почему сегодня ставится задача еще более важная, чем массовый выпуск машин - к о м п ь ю т е р н а я г р а м о т н о с т ь всего населения страны.

§1. ЧТО ТАКОЕ ИНФОРМАТИКА?

И н ф о р м а т и к а - наука о законах и методах накопления, передачи и обработки информации с помощью вычислительных машин.

Под информацией будем понимать совокупность сведений о предметах и явлениях окружающего мира, предназначенных для хранения, передачи и обработки. Информация может выражаться числами символами, графиками, таблицами, сигналами и др.

Когда требуется решить некоторую задачу, то обычно задают множество D, называемое множеством данных, множество R, называемое множеством возможных результатов и некоторый закон f, по которому данные преобразуются в результат.



О б р а б о т к а и н ф о р м а ц и и есть практическая реализация функции f, отображающей D в R. При обработке информации обычно происходит ее сокращение (сужение).

П р и м е р . Дан треугольник $\triangle ABC$ со сторонами длиной a, b и c. Требуется вычислить его площадь S.

Здесь D - это сведения о том, что плоская фигура, площадь которой отыскивают - треугольник, и конкретные значения a, b и c. Множество R содержит одно значение - S. Функция f может быть задана несколькими способами, в зависимости от исполнителя (решателя) задачи. В частности, может вообще не быть задана, в предположении, что исполнителю известна последовательность действий, ведущая от исходных данных к результату.

Пусть сказано, что площадь S требуется вычислить по формуле Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где p - полупериметр треугольника.

Дальнейшая детализация действий опять зависит от потенциального исполнителя (возможно, он не знает, что такое полупериметр ?). В общем случае требуется точная последовательность предписаний (указаний, инструкций, команд), которая преобразует множество D во

множество R .

Рассмотрим еще один пример. Пусть требуется вычислить корни квадратного уравнения

$$ax^2 + bx + c = 0 \quad (a \neq 0)$$

Здесь тоже можно указать несколько вариантов обработки исходных данных, ведущих к результату; например, для уравнения

$$x^2 - 5x + 6 = 0$$

отдельные исполнители представят уравнение в виде

$$x^2 - 5x + 6 = (x-2)(x-3)$$

и выдадут ответ : $x_1 = 2$; $x_2 = 3$.

Более общим является следующая последовательность предписаний (инструкций , команд) :

Шаг I. прочитать числа a , b , c ;

Шаг 2. вычислить $D = b^2 - 4ac$;

Шаг 3. если $D < 0$, перейти к шагу 6, иначе - к шагу 4;

Шаг 4. вычислить $x_1 = (-b+\sqrt{D})/2a$, $x_2 = (-b-\sqrt{D})/2a$;

Шаг 5. записать x_1 , x_2 ; перейти к шагу 7;

Шаг 6. записать: "действительных корней нет";

Шаг 7. конец.

Если такая последовательность составлена, то выполняя ее команды I-7, можно действовать а в т о м а т и ч е с к и или вообще попытаться передать решение этой задачи некоторому исполнителю - автомату, в частности, ЭВМ.

Приведенная выше последовательность инструкций называется а л г о р и т м о м . А л г о р и т м решения задачи - точное предписание о последовательности действий, которые должны быть произведены над исходными данными для получения результата решаемой задачи. Это понятие является важнейшим в информатике и в последующем мы к нему еще не раз будем обращаться.

§ 2. ПЕРВОНАЧАЛЬНЫЕ СВЕДЕНИЯ ОБ ЭВМ

Родословная современных вычислительных машин берет начало в седой древности, когда счет производился с помощью пальцев на руках, названных Ф.Энгельсом "своеобразной счетной машиной". А потом, начиная с XVII века, когда появились первые механические счетные машины, эта техника непрерывно совершенствовалась и наконец обогатились идеей программирования. Произошло это, когда крупнейший английский физик Чарльз Бэббидж создал проект аналитической машины с программным управлением.

Электронные вычислительные машины (ЭВМ) предназначены для автоматического решения разнообразных трудоемких задач, возникающих в различных областях человеческой деятельности. Применение ЭВМ совершенно необходимо там, где решение задач с помощью других средств или "ручным" способом требует затраты большого времени и усилий вычислителей, либо вообще не может быть получено.

ЭВМ - быстродействующее ц и ф р о в о е п р о г р а м м н о у п р а в л я е м о е э л е к т р о н н о е устройство для обработки информации (рис. 2.I).

Рис. 2.I

Слово "электронное" означает, что любое действие по обработке и перемещению информации с момента ее ввода и до вывода результатов осуществляется посредством электрических сигналов, передаваемых внутри ЭВМ со скоростью приблизительно равной скорости прохождения электрического тока по проводам.

Обратимся к ранее записанной последовательности инструкций для решения квадратного уравнения. Такую последовательность можно назвать п р о г р а м м о й вычислений. Можно сказать, что ЭВМ - это высоконадежные, быстродействующие и точные а в т о м а т ы , способные выполнять программы, составленные человеком в

виде, "понятном" для машины. Характерная особенность современных ЭВМ - они запоминают программу вычислений. Человеку остается ее составить и поместить в память машины, остальное она сделает автоматически. В этом суть программно-управляемых машин.

Программа для ЭВМ, как и программа действий для человека, состоит из отдельных команд. Команда - это указание вычислительной машине на выполнение элементарных действий над данными, таких, например, как сложение, вычитание, умножение, деление.

Формы представления информации в ЭВМ

ЭВМ может хранить и обрабатывать информацию в виде комбинации электрических сигналов двух типов, которые принято обозначать цифрами 0 и I. В ЭВМ любая информация представляется последовательностью этих двух цифр. В большинстве современных ЭВМ один символ (буква, цифра, знак препинания, специальный знак §, % и т.д.) записывается (кодируется) с помощью 8 двоичных цифр 0 и I. Например, буква А кодируется как II00 0001, а буква М - II0I 0I00, так что слово МАМА кодируется последовательностью из 32 цифр.

II0I 0I00 II00 0001 II0I 0I00 II00 0001

Поскольку многие важнейшие применения ЭВМ связаны с обработкой числовой информации, то числа кодируются последовательностями двоичных цифр; например, число 25 - последовательность из I6 цифр:

0000 0000 000I I00I

Каждый двоичный разряд называется битом; восемь бит - единица информации, называемая байтом. Байт принят за единицу измерения объема памяти современных ЭВМ. Это минимальная адресуемая единица памяти ЭВМ (порядковый номер байта ОЗУ - это его адрес). Более крупные единицы памяти

I Кбайт = I Килобайт = 2^{10} байтов = I024 байтов

I Мбайт = I Мегабайт = 2^{10} Кбайт

I Гбайт = I Гигабайт = 2^{10} Мбайт.

Структура ЭВМ и ее работа

Процессу автоматической обработки информации (а решение задач с помощью ЭВМ представляет собой именно такой процесс) свойственны следующие действия :

- ввод (чтение) начальной информации;
- хранение (запоминание) информации;
- обработка информации (собственно решение задачи);
- управление процессом обработки;
- вывод (запись) результатов.

В ЭВМ эти действия обеспечиваются с помощью следующих устройств:

I. Запоминающее устройство или память : различают оперативное запоминающее устройство (ОЗУ) и внешнее запоминающее устройство (ВЗУ) (в программируемых микрокалькуляторах - постоянное запоминающее устройство - ПЗУ);

2. Обрабатывающее, или арифметико-логическое устройство (АЛУ);

3. Устройство управления (УУ);

4. Устройство ввода (Ввод);

5. Устройство вывода (Вывод) .

Схема ЭВМ и связи между ее компонентами представлены на рис. 2.2. Совокупность УУ и АЛУ называют также процессором ЭВМ.

Приведем краткую характеристику составных частей ЭВМ.

Устройство управления координирует работу всех устройств ЭВМ, управляет потоками информации внутри системы. Действия УУ определяются командами программы, которая находится в ОЗУ. УУ просматривает программу и выполняет ее по одной команде : расшифровывает код операции команды, вызывает из ОЗУ участвующие в операции числа и отправляет их в АЛУ, а затем пересылает в память полученный результат.

Запоминающее устройство служит для хранения всей необходимой информации, сопутствующей процессу решения задачи: исходных, промежуточных и результирующих данных, а также самой программы, т.е. алгоритма счета, записанного в специальном виде, "понятном" машине.

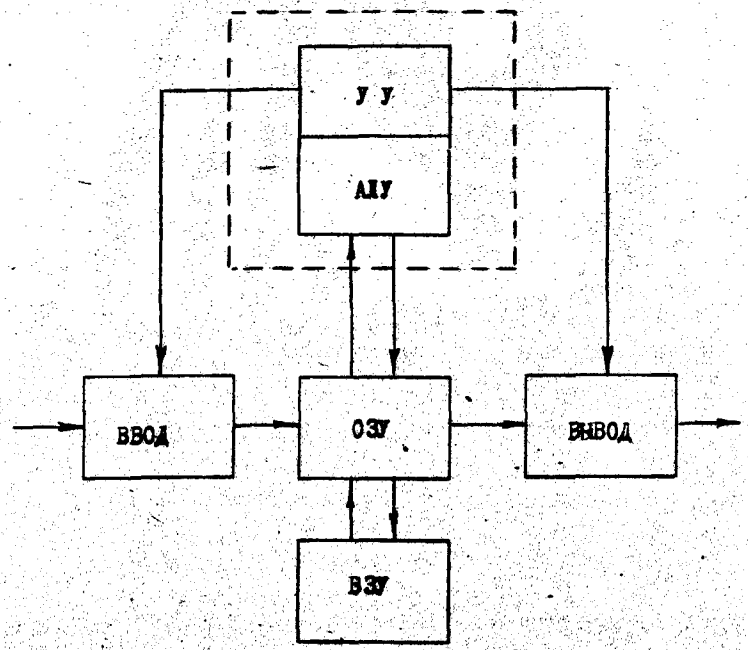


Рис. 2.2

ОЗУ можно рассматривать как набор ячеек (подобно ячейкам автоматической камеры хранения), предназначенных для хранения единиц информации. Каждая ячейка имеет свой номер, называемый ее адресом. Следует четко различать адрес ячейки и содержимое ячейки. К содержимому каждой ячейки можно обратиться с помощью ее адреса.

Арифметико-логическое устройство служит для выполнения над информацией арифметических операций, операций сравнения. Под руководством УУ информация попадает в АЛУ вместе с управляющими сигналами, указывающими, какие действия надо выполнять и куда поместить результат. АЛУ способно выполнять только те операции, которые определены конструкцией данной ЭВМ. Полная совокупность таких операций образует с и с т е м у к о м а н д (операций) ЭВМ.

Устройства ввода и вывода осуществляют связь человека с ЭВМ. Дело в том, что представление информации внутри ЭВМ сильно отличается от общепринятого представления ее среди людей: там она, как уже отмечалось, представлена в виде электрических сигналов - именно в этой форме и производится ее обработка в АЛУ. Устройства ввода-вывода выполняют роль преобразователей информации от внешнего представления ко внутреннему (машинному) и наоборот.

Для подготовки информации (программ и данных) к вводу их в ЭВМ традиционно используют перфокарты . Для подготовки на перфокартах необходимой информации используется специальное устройство подготовки данных, которое преобразует элементарные символы (буквы, цифры, специальные знаки), представленные на его клавиатуре, в в двоичный код и пробивает в карте отверстия, соответствующие двоичному представлению каждого очередного символа.

На рис. 2.3 изображена перфокарта. Закодированные символы напечатаны вдоль верхнего края карты. Каждая из 80 колонок карты представляет один символ; код каждого символа расположен вертикально под его печатным изображением.

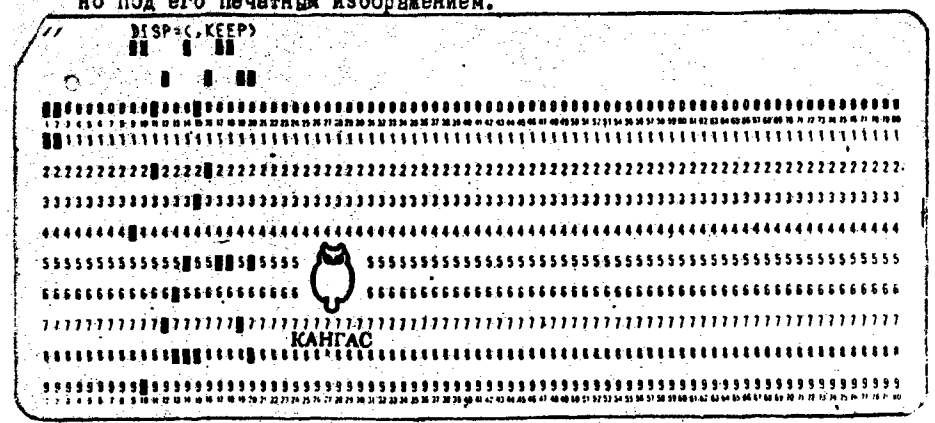


Рис. 2.3 Перфокарта с информацией

Устройство чтения перфокарт содержит электромеханические или оптические датчики, определяющие, в каких строках и столбцах пробиты отверстия. Устройство читает карты по очереди, переводя закодированные символы в их двоичное представление и вводит их в память ЭВМ. Наиболее широко используемым устройством вывода информации в ЭВМ является п е ч а т а ю щ е е у с т р о й с т в о,

прежде всего потому, что напечатанная информация может быть легко использована. Печатающее устройство преобразует информацию, полученную от устройства управления в двоичной форме, в символьную и печатает полученный текст, по одной строке; в результате получается текст, очень напоминающий текст, напечатанный на пишущей машинке.

Кроме рассмотренных выше устройств, используется и множество других. Большие удобства пользователям предоставляет т е р м и н а л ь н о е устройство с электронно-лучевой трубкой, напоминающее телевизор и получившее название д и с п л е й. Дисплеи имеют клавиатуру и с них, как с пишущей машинки, можно вводить программу и данные в ЭВМ, только в этом случае набранный текст не печатается, а высвечивается на экране. Информация, которая выводится из машины, также подается на экран. Она может иметь форму либо алфавитно-цифрового текста, либо графического материала. Это делает работу с дисплеем особенно удобной.

С помощью м а г н и т н ы х л е н т и д и с к о в можно организовать простой и быстрый ввод и вывод, а также хранение больших количеств информации.

Типичный процесс обработки данных на ЭВМ выглядит следующим образом:

Программа и исходные данные считываются устройством ввода, которое заносит их в ОЗУ. УУ пересылает информацию, подлежащую обработке, в АЛУ и выполняет над ней необходимые арифметические или логические операции. Результаты пересылаются обратно в ОЗУ. Если в соответствии с программой требуется вывод информации, то окончательно результат передается на устройства вывода.

Понятие о программном (математическом) обеспечении ЭВМ

Истинная ценность современной ЭВМ – это не столько аппаратура, из которой она состоит, сколько совокупность написанных для нее программ, расширяющих возможности машины по обработке информации и имеющих статус многократно используемых. Можно сказать, что ЭВМ способна воспринимать и накапливать знания. В самом деле, написать для некоторой ЭВМ, например, программу вычисления корней квадратного уравнения – это значит научить машину решать квадратные уравнения. Теперь можно сколько угодно предлагать этой ЭВМ

различные квадратные уравнения и она будет их решать по заданной программе. Совокупность таких программ для решения различных научно-технических, хозяйственных и других задач, оформленных специальными образом, составляет п р и к л а д н о е п р о г р а м м н о е о б е с п е ч е н и е ЭВМ. В последнее время такие функционально завершенные комплексы программ вместе с эксплуатационной документацией, предназначенные для решения задач некоторой предметной области, называют п а к е т а м и п р и к л а д н ы х п р о г р а м м. Программы, входящие в пакет, выполняются не отдельно друг от друга, а совместно, во взаимосвязи, диктуемой спецификой решаемой задачи. Разрабатывать программы, составляющие пакет, то есть передавать ЭВМ систематизированные знания, труднее, чем разрабатывать разрозненные, независимые программы.

Современная ЭВМ – сложный многофункциональный комплекс, состоящий из большого числа различных взаимодействующих устройств, и управление ими представляет собой весьма сложную задачу. Решением этой задачи занимается специальная система программ, называемая о п е р а ц и о н н о й с и с т е м о й. Помимо управления работой ЭВМ, операционная система выполняет и ряд других функций, важнейшая из которых – это пополнение набора системы команд ЭВМ новыми операциями, весьма удобными для программистов, но слишком сложными для их аппаратурной реализации на ЭВМ.

Кроме операционной системы, на ЭВМ отдельно выделяют систему программ, предназначенных для автоматизации самого процесса разработки программ. Этот комплекс программ называется с и с т е м о й п р о г р а м м и р о в а н и я и включает, в частности, так называемые т р а н с л я т о р ы – программы, обеспечивающие перевод программ с различных языков, на которых программисты составляют программы, на единый для данной ЭВМ машинный язык.

Операционная система ЭВМ, ее система программирования и комплекс системных обрабатывающих и обслуживающих программ, вместе с которыми поставляются ЭВМ пользователю, образуют ее с и с т е м н о е (стандартное) п р о г р а м м н о е о б е с п е ч е н и е. Поскольку первые ЭВМ применялись почти исключительно для математических расчетов, за программным обеспечением закрепилось также название м а т е м а т и ч е с к о е о б е с п е ч е н и е.

Чтобы эффективно и грамотно использовать современную ЭВМ, пользователь должен хорошо знать структуру и состав ее системного

программного обеспечения и те возможности, которые оно предоставляет для разработки прикладных программ.

§ 3. ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА ЭВМ. ПОНЯТИЕ О МАТЕМАТИЧЕСКОМ МОДЕЛИРОВАНИИ

Ранее было установлено, что ЭВМ работает под управлением программы, состоящей из последовательности команд. Разработка программ для процессора ЭВМ составляет предмет п р о г р а м м и р о в а н и я в узком смысле слова. В широком же смысле программирование предполагает выполнение целого ряда последовательных действий, связанных с подготовкой и решением задач с помощью вычислительных машин. В общем случае процесс решения задачи на ЭВМ включает в себя следующие этапы.

1. Постановка задачи.
2. Построение математической модели.
3. Разработка алгоритма задачи.
4. Составление программы для ЭВМ.
5. Реализация программы на ЭВМ.
6. Анализ результатов.

Постановка задачи описывает сущность решаемой проблемы и выполняется с привлечением языка той предметной области (экономики, медицины, геологии и т.д.), для которой ставится задача. В постановке должны быть четко оговорены входные, промежуточные и выходные данные. Нередко в постановке сложных народнохозяйственных задач участвуют, кроме специалистов данной отрасли, также специалисты по информатике. Поэтому уже при постановке задачи выполняется ее формализация, вводится математическая терминология и обозначения.

Построение математической модели задачи состоит в том, что выделяются наиболее существенные черты и свойства исследуемой проблемы и записываются в виде некоторых математических зависимостей. Математическая модель основана на некотором упрощении, идеализации реального объекта или явления и потому не тождественна (или, как говорят, не адекватна) объекту. Однако, использование математической модели позволяет свести исследование реального процесса к решению математической задачи, открывая тем самым возможность применения для его изучения методов прикладной математики

и современных ЭВМ. После этого наступает следующий этап исследования – поиск метода решения сформулированной математической задачи.

Пусть требуется вычислить площадь приусадебного земельного участка. Предполагая, что он имеет форму прямоугольника, измеряют его длину и ширину, а затем перемножают полученные числа. Здесь мы по существу реальный объект (земельный участок) заменили его математической моделью – прямоугольником. Прямоугольнику приписываются размеры, полученные в результате измерения и площадь такого прямоугольника приближенно принимается за искомую площадь участка.

В действительности же форма земельного участка редко представляет собой прямоугольник и замена ее прямоугольником и есть та идеализация, огрубление, о которых говорилось выше. В зависимости от полученных результатов и требований точности, исходная модель может уточняться в процессе решения задачи.

Рассмотрим следующий пример. На железнодорожные станции A_1 и A_2 прибыло 30 вагонов с мукой, по 15 вагонов на каждую. Все вагоны требуется доставить на хлебозаводы B_1 и B_2, потребности которых – соответственно 10 и 20 вагонов. Затраты на транспортировку одного вагона из пункта A_1 в пункты B_1 и B_2 – 1 и 3 руб. соответственно, а из A_2 в B_1 и B_2 – 2 и 5 руб. соответственно. Составить такой план транспортировки, чтобы суммарные затраты были минимальны.

Это пример постановки экономической задачи. Для ее решения построим математическую модель. Обозначим через x_{ij} количество вагонов, перевозимых из пункта A_i в пункт B_j (i =1,2; j = 1,2). Сведем все данные в следующую таблицу.

Таблица 3.1

	B_1	B_2	
A_1	1 x_{11}	3 x_{12}	15
A_2	2 x_{21}	5 x_{22}	15
	10	20	

В этой таблице числа 15 и 15 в последнем столбце указывают число вагонов, имеющихся в пунктах A_1 и A_2, числа 10 и 20 последней строки – потребности хлебозаводов B_1 и B_2 соответственно; числа 1, 3, 2, 5, обозначенные в правых верхних углах клеток

– стоимости перевозки одного вагона от A_i поставщика к B_j потребителю ($i = 1,2$; $j = 1,2$). Исходя из экономического смысла постановки задачи, имеем:

$$x_{11} + x_{12} = 15$$
$$x_{21} + x_{22} = 15 \qquad (3.1)$$

Эти равенства указывают, что все вагоны из станций A_1 и A_2 должны быть вывезены. С другой стороны, хлебозаводы должны обязательно получить требуемое количество вагонов, т.е.

$$x_{11} + x_{21} = 10$$
$$x_{12} + x_{22} = 20 \qquad (3.2)$$

Нетрудно видеть, что

$$x_{ij} \geq 0 \quad (i=1,2;\ j=1,2) \qquad (3.3)$$

т.е. все x_{ij} – целые неотрицательные числа.

Общую стоимость F транспортировки можно выразить соотношением:

$$F = x_{11} + 3x_{12} + 2x_{21} + 5x_{22} \qquad (3.4)$$

Таким образом, решение исходной задачи сводится к нахождению таких значений x_{11}, x_{12}, x_{21}, x_{22}, удовлетворяющих соотношениям (3.1) – (3.3), при которых функция F достигает наименьшего значения. На этом построение математической модели заканчивается и начинается следующий этап – построение алгоритма решения задачи. Как уже отмечалось, под алгоритмом понимают конечное упорядоченное множество предписаний (указаний, команд) исполнителю по выполнению последовательности действий, направленных на решение поставленной задачи. В рассмотренном примере алгоритм может быть записан так :

1. Выразить x_{12}, x_{21}, x_{22} из соотношений (3.1) – (3.2) через x_{11}.
2. Подставить полученные выражения для x_{12}, x_{21}, x_{22} в выражение (3.4) для функции F.
3. Найти x_{11}, при котором функция F достигает минимального значения.
4. Используя найденное x_{11}, вычислить остальные значения неизвестных x_{12}, x_{21}, x_{22}.

При исполнении алгоритма оказывается, что решение задачи сводится к исследованию функции одной переменной

$$F = x_{11} + 90$$

Учитывая (3.3), ясно, что при $x_{11} = 0$ функция F принимает наименьшее значение. Таким образом, оптимальный план перевозки вагонов следующий :

$$x_{11} = 0; \quad x_{12} = 15; \quad x_{21} = 10; \quad x_{22} = 5.$$

Из таблицы 3.1 видно, что из станции A_1 все вагоны следует отправить в пункт B_2, из станции A_2 10 вагонов надо отправить в пункт B_1, а 5 – в B_2. Транспортные расходы при этом составят 90 руб.

Интересно отметить, что оптимальный план перевозки построен без использования самого выгодного на первый взгляд маршрута A_1B_1. Рассмотренная выше задача относится к классу так называемых транспортных задач, имеющих огромное практическое значение. При наличии большого числа поставщиков и потребителей (исчисляемого, например, сотнями), эта задача практически неразрешима с помощью предложенного выше метода и для ее решения на ЭВМ применяются другие, более сложные алгоритмы.

Следующий этап решения задачи – составление программы для ЭВМ. Программа представляет собой алгоритм решения задачи, но реализованный в виде, "понятном" вычислительной машине, на специальном языке программирования. Применение ЭВМ для решения различных задач предъявляет очень строгие требования к последовательности выполняемых действий алгоритмов и точности описания правил. Поэтому языки программирования строго формализованы, чтобы не допускать при описании алгоритмов никаких неопределенностей, неясностей в правилах и последовательности выполняемых действий. Этап исполнения программы на ЭВМ завершается в конечном итоге получением результатов решения. Наконец, завершающий этап решения анализ результатов решения задачи. Если полученные результаты не соответствуют ожидаемым (контрольным) расчетам, происходит возврат на один из предыдущих этапов, где возможны модификация программы, уточнение алгоритма решения и даже пересмотр исходной постановки задачи.

Рассмотренные в этом параграфе основные понятия, связанные с обработкой данных на ЭВМ, будут в дальнейшем развиваться и уточняться, причем, основное внимание будет уделено построению алгоритмов для решения задач.

§ 4. АЛГОРИТМЫ И ИХ СВОЙСТВА

В большинстве задач школьного курса математики и физики ответ дается в виде готовой формулы, определяющей последовательность математических операций, которую надо выполнить для вычисления искомой величины. Например, формула Герона выражает площадь треугольника через длины его сторон, закон Ома связывает функциональной зависимостью напряжение, величину тока и сопротивление в электрической цепи и т.д.

Однако даже в математике и физике, не говоря уже о других науках, встречаются задачи, решение которых не удается получить в виде формулы, связывающей искомые величины с заданными. Тем не менее ответ для них может быть найден, если задана последовательность действий, которую нужно выполнить для достижения цели. Вспомним ,например, известное из младших классов правило вычисления суммы нескольких многозначных чисел с помощью поразрядного сложения "столбиком". Пользуясь этим правилом, разные люди выполняют одинаковые действия, в одной и той же последовательности и получают независимо друг от друга один и тот же результат, если исходные данные были одинаковы. Одна из особенностей рассматриваемого правила состоит в том, что оно применимо не для какой-то конкретной пары чисел, а для произвольных слагаемых.

Как уже отмечалось, понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение указанной цели или на решение поставленной задачи, называют а л г о р и т м о м . Чтобы лучше уяснить суть этого понятия, рассмотрим решение нескольких задач.

Выбор наибольшего из трех чисел x , y , z .

Для решения этой задачи применима следующая последовательность действий : сначала сравним между собой два первых числа x и y и в качестве промежуточного результата t выберем большее из них (в случае их равенства берем любое, например, x). Полученное таким образом значение t далее сравниваем с z и опять выбираем наибольшее. Результат этого действия и будет окончательным решением задачи.

Отыскание наибольшего общего делителя (НОД) двух целых положительных чисел m и n .

Общей формулы для решения этой задачи, как известно, не существует. Однако, можно указать универсальные методы, которые позволяют найти НОД любых натуральных чисел.

Один из таких методов заключается в последовательном переборе чисел n , $n-1$, $n-2$ и т.д. (для определенности предположим, что $m > n$). Процесс продолжается до тех пор, пока не обнаружится число, являющееся одновременно делителем m и n . Такая последовательность действий всегда приводит к цели, хотя вряд ли нужно говорить о трудоемкости и неэффективности этого метода.

Рассмотрим другой, более совершенный алгоритм. Разделим m на n в целых числах с остатком. Пусть получены частное q и остаток r ($0 \le r < n$). Это означает, что число представимо в виде

$$m = nq + r \qquad (4.1)$$

Если остаток r равен нулю, то число n - НОД пары чисел (m , n) и задача решена. В противном случае из (4.1) вытекает следующее утверждение : всякий общий делитель чисел m и n является одновременно общим делителем чисел n и r , и наоборот.

Из этого утверждения следует, что далее надо искать НОД чисел n и r . Для этого снова разделим n на r в целых числах с остатком и к ним, как и к исходным числам m и n , применим приведенные выше рассуждения. В результате конечного числа шагов процесс обрывается, последний делитель, для которого $r = 0$, и будет искомым НОД. Эта последовательность действий для нахождения НОД известна с глубокой древности как а л г о р и т м Е в к л и д а . Запишем его в более компактной форме.

1. Разделить m на n ; пусть остаток равен r .
2. Если остаток равен нулю, то перейти к шагу 4, иначе - к шагу 3 .

3. Заменить m на n, а n на r, перейти к шагу I.
4. Принять за НОД последнее полученное значение n.

Лучший способ понять алгоритм – это испытать его с бумагой и карандашом в руке. Для примера разберем алгоритм Евклида для $m = 14$, $n = 21$. Начнем с шага I. Деление m на n в данном случае осуществляется очень просто, поскольку частное равно нулю; остаток же равен I4. Переходим к шагу 2. Поскольку $r \neq 0$, то в шаге 3 новое значение m становится равным 21, а n – I4 и процесс начинается с самого начала. Отсюда ясно, что если с самого начала $m < n$, то алгоритм приводит ко взаимному обмену значений m и n (обладает элементарным самообучением !). При желании мы могли бы добавить новый шаг к алгоритму:

0. Если $m < n$, то обменять их местами.

Возвращаясь к шагу I, находим, что остаток от деления равен 7. Второй шаг снова отсылает нас к пункту 3, где m становится равным I4, а n – 7. Следующий цикл дает на первом шаге $r = 0$, а шаг 2 устанавливает конец алгоритма, передав управление шагу 4. НОД равен последнему значению n, то есть 7.

И Г Р А Б А Ш Е. Эта игра заключается в следующем. На столе разложены II предметов (можно брать I5, I9, вообще $(4k-I)$ предметов, $k = 1,2,\ldots$). Играют двое. Соперники ходят по очереди, за каждый ход любой из играющих может взять I,2 или 3 предмета. Проигрывает тот, кто вынужден взять последний предмет.

Анализ игры показывает, что первый игрок всегда будет выигрывать, если он будет руководствоваться следующими указаниями.

I. Взять два предмета.
2. Предоставить ход сопернику; запомнить количество k взятых им предметов при последнем ходе.
3. Если к очередному ходу первого игрока еще остались неотобранные предметы, то взять $4 - k$ предметов; перейти к шагу 2.
4. Объявить о выигрыше.

Последний пример показывает, что понятие алгоритма используется не только при решении задач вычислительного характера. Это понятие в широком смысле предполагает, что исходными данными и результатами алгоритма могут служить самые различные объекты. Это открывает возможность широкого применения этого понятия. Можно говорить об алгоритмах изготовления детали на токарном станке, перевода с одного языка на другой, пеленания ребенка, приготовления кофе, разборки и сборки армейского автомата и т.д. Можно с уверенностью сказать, что даже не употребляя слова "алгоритм", многие интуитивно пользуются этим понятием и правильно представляют его смысл. Алгоритмический подход, обращение к "бытовым" алгоритмам неотделимы от повседневной жизни людей, от их обычной работы. В подавляющем большинстве случаев результат деятельности человека прямо зависит от того, насколько четко он чувствует алгоритмическую сущность своих действий : **ч т о** делать в каждый момент, в **к а к о й** последовательности, **к а к и м** должен быть результат и т.п.

Не могут быть оставлены без внимания принципы алгоритмизации и в процессе обучения, в частности, в преподавании школьных дисциплин. Хорошо известно, что алгоритмы буквально пронизывают содержание всех школьных предметов, так что само содержание школьных наук при соответствующей его подаче предоставляет большие возможности для систематического формирования алгоритмических навыков. При этом умелое введение ученика в алгоритмическую природу понятий, с одной стороны, помогает более четкому усвоению изучаемых предметов и явлений окружающего мира, а с другой стороны, подготавливает его к последующему восприятию основных идей использования программно- управляемой вычислительной техники.

Приведем в качестве иллюстрации задачу построения биссектрисы заданного угла ABC (угол меньше 180^0)из школьного курса геометрии, применяя для этого следующий алгоритм.

I. Установить ножку циркуля в точку B.
2. Провести окружность произвольного радиуса R.
3. Отметить точки M и N пересечения окружности со сторонами BA и BC.
4. Провести окружность с центром в точке M и радиусом R.
5. Провести окружность с центром в точке N и тем же радиусом.
6. Отметить точку P пересечения построенных окружностей с центрами в точках M и N.
7. Провести луч BP. BP – биссектриса $\angle ABC$.
8. Конец.

Понятие алгоритма возникло и используется значительно раньше появления ЭВМ, однако, широкое распространение это понятие получило благодаря идее *автоматизации* поведения исполнителя-автомата, реализуемой на основе алгоритма. В ряду всевозможных автоматов ЭВМ является наиболее впечатляющим примером такого исполнителя. Реализуя алгоритм, исполнитель может не вникать в смысл того, что он делает и вместе с тем получать правильный результат. В таком случае говорят, что исполнитель действует *формально*, т.е. отвлекается от содержания поставленной задачи и только строго выполняет команды алгоритма в указанной последовательности. Во многих случаях выполнение алгоритма можно поручить не человеку, а машине. Это возможно, если простейшие операции, на которые при создании алгоритма расчленяется процесс решения задачи, машине понятны и она может их реализовать. Это накладывает на запись алгоритмов целый ряд требований, которые мы сформулируем в виде перечня свойств, которым должны удовлетворять алгоритмы.

Свойства алгоритмов

I. Каждый алгоритм разбивает весь процесс на отдельные автономные шаги и должен содержать исчерпывающую информацию как о тех действиях, которые нужно выполнить на каждом из них, так и о том порядке, в котором должны выполняться эти шаги. В результате такого разбиения запись алгоритма представляет упорядоченную совокупность предписаний (команд, директив), образующих *прерывную* (дискретную) структуру. Только выполнив требования одной команды, можно приступать к выполнению следующей.

Это свойство алгоритмов называется *дискретностью*.

Составим алгоритм выбора наибольшего из трех заданных чисел x, y, z. При решении этой задачи можно руководствоваться следующей системой предписаний.

1. Сравнить x с y. Если $x > y$, то принять $t = x$, иначе $t = y$.
2. Сравнить t с z. Если $z > t$, то принять $t = z$.
3. Принять t в качестве результата. Закончить работу.

Дискретная структура алгоритмической записи в данном случае подчеркивается сквозной нумерацией отдельных предписаний алгоритма. Если при исполнении текущего предписания никак не определяется номер очередного предписания, то по умолчанию предполагается переход к предписанию, следующему за данным в порядке возрастания номеров.

2. Алгоритм должен всегда заканчиваться после конечного числа шагов. Это свойство алгоритмов называется *конечностью* (финитностью). Так, рассмотренный ранее алгоритм Евклида (стр. 24) удовлетворяет этому условию, так как после шага I значение r меньше, чем n, поэтому при $r \neq 0$ значение n уменьшается к следующему выполнению шага I. Убывающая последовательность положительных целых чисел должна в конце концов закончиться, поэтому алгоритм выполняется только конечное число раз для любого первоначального значения n.

Евклид предложил также некоторую вычислительную процедуру для получения *наибольшей общей меры* двух отрезков, которая по сути эквивалентна приведенному выше алгоритму нахождения наибольшего общего делителя. Однако эта процедура не может считаться алгоритмом, так как в случае *несоизмеримых* отрезков она никогда не заканчивается.

3. Каждый шаг алгоритма должен быть точно определен. Это означает, что он должен быть представлен в такой форме, чтобы любые исполнители понимали и выполняли бы его совершенно одинаково. Это свойство алгоритма называется *определенностью* или *детерминированностью*.

В данном случае речь идет о том, что запись алгоритма должна быть настолько точной и полной, чтобы у исполнителя не возникало потребности в принятии каких-либо самостоятельных решений, не предусмотренных составителем алгоритма. Нередко в качестве примеров алгоритмов приводят кулинарные рецепты. Однако, те из них, которые содержат предписания типа "добавить щепотку соли", "взять 2-3 столовые ложки масла" - не могут выступать в качестве алгоритмов из-за недостаточной определенности своих предписаний.

Словесное описание алгоритмов не всегда обеспечивает их определенность. Чтобы преодолеть эту трудность, для описания алгоритмов разработаны формально определенные *алгоритмические языки* и *языки программирова-*

н и я , в которых каждое утверждение имеет абсолютно точный смысл. Запись алгоритма на языке программирования называется п р о г р а м м о й для ЭВМ.

4. От алгоритма требуют также, чтобы он был э ф ф е к т и в н ы м. Это означает, что все действия, которые необходимо провести в алгоритме, должны быть достаточно простыми, чтобы их в принципе можно было выполнить с помощью ручного счета.

5. Алгоритмы редко разрабатываются для решения какой-либо конкретной задачи, чаще – для широкого круга задач данного типа. Во многих случаях это удается сделать за счет использования в алгоритме не конкретных числовых величин, а переменных, значения которых вводятся в алгоритм извне во время его исполнения, причем, сам алгоритм должен учитывать различные возможные значения входных данных и их комбинации. Так, алгоритм вычисления корней квадратного уравнения $ax^2+bx+c=0$ разрабатывается не для одного конкретного уравнения, а для уравнений с любыми значениями a, b, c. Именно поэтому такой алгоритм должен в общем случае содержать тщательный анализ значений как самих коэффициентов, так и их комбинаций (в частности, проверку условия $b^2-4ac\geq 0$). Эту особенность алгоритмов называют свойством м а с с о в о с т и .

§ 5. СПОСОБЫ ОПИСАНИЯ АЛГОРИТМОВ

Алгоритмы можно описывать многими различными способами. В каждом отдельном случае выбор средств и методов для о п и с а н и я алгоритма зависит прежде всего от природы самого алгоритма и, что наиболее существенно, от того, кто будет исполнителем алгоритма – человек или машина.

Рассмотрим некоторые способы задания алгоритмов, рассчитанных на исполнителя-человека.

5.1. Формулы

Простейший класс алгоритмов составляют алгоритмы вычислений по математическим формулам, которые с учетом общепринятых правил их написания и исполнения представляют собой некоторый алгоритмический язык.

Строго говоря, формула определяет последовательность действий не столь однозначно, как того требует понятие алгоритма (в частности, свойство определенности). Например, формула

$$S = V_0 t + \frac{at^2}{2}, \qquad (5.1)$$

задающая вычисление пути, пройденного телом за время t в зависимости от начальной скорости V_0 и ускорения a при равнопеременном движении, допускает определенный произвол в порядке проведения вычислений. Молчаливо предполагается, что исполнитель обучен установленным в математике порядку и правилам вычислений, так что выбор любого порядка действий не отразится на результате.

В вычислениях по формулам следует уделять особое внимание наиболее рациональным способам, преобразуя, если нужно, формулы к виду, наиболее удобному для вычислений. Так, последняя формула, несмотря на ее простоту, допускает преобразование к виду

$$S = \left(V_0 + \frac{at}{2}\right) t, \qquad (5.2)$$

которая более рациональна с точки зрения вычислений, чем (5.1). Вообще, вычисление значения многочлена

$$a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n$$

требует $2n-1$ умножений и n сложений, в то время как эквивалентная форма

$$((\ldots(a_0x + a_1)x + a_2)x + \ldots + a_{n-1})x + a_n$$

требует уже только n умножений и n сложений.

От умения рационально программировать линейные алгоритмы часто зависит эффективность всей программы для ЭВМ.

5.2. Табличный способ

Запись вычислительного алгоритма в форме таблицы особенно удобна при многократных вычислениях по одной и той же формуле. Таблица представляет разбивку формулы на последовательность элементарных действий, обеспечивающих четкую последовательность предписаний и пооперационную регистрацию результатов. Таблица фиксирует входные данные (аргументы), промежуточные и выходные данные (результаты). Например, для ранее рассмотренной формулы (5.2) табличный алгоритм может иметь вид.

Таблица 5.I

V_0	a	t	at	$\frac{at}{2}$	$V_0+\frac{at}{2}$	S
2.6	I.4	3.0	4.2	2.I	4.7	I4.I
I.7	2.9	3.0	8.7	4.35	6.05	I8.I5
9.3	0.6	3.0	I.8	0.9	I0.2	30,6

Табличные алгоритмы и формулы широко применяются при вычислениях с помощью микрокалькуляторов. Появился даже своеобразный алгоритмический язык, включающий, кроме входящих в формулу параметров, обозначения клавиш микрокалькулятора, которые необходимы для получения требуемого результата. Например, программа вычислений пройденного телом пути S по формуле (5.2) на микрокалькуляторе "Электроника Б3-I9М" для различных значений аргументов V_0, a, t может быть записана следующим образом :

$$a\ \boxed{+}\ t\ \boxed{\times}\ 2\ \boxed{\div}\ V_0\ \boxed{+}\ t\ \boxed{\times}$$

С помощью формул и таблиц удобно записывать алгоритмы, последовательность вычислительных действий в которых не зависит от конкретных значений входящих в них величин и определена самой структурой алгоритма. Такие алгоритмы называются л и н е й н ы м и .

Однако формулы могут определять алгоритмы, для однозначного описания которых линейных алгоритмов оказывается недостаточно. Рассмотрим, например, алгоритм вычисления функции, заданной графически (рис. 5.I).

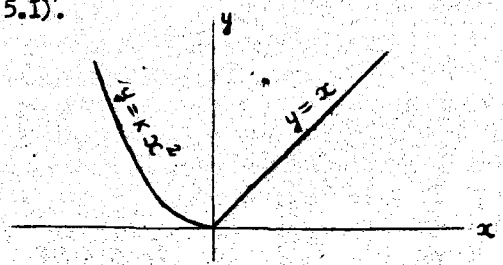


Рис. 5.I.

Такой алгоритм уже не является линейным, так как в нем заложена операция в ы б о р а одной из вычислительных формул в зависимости от заданного значения x :

$$y=\begin{cases} kx^{l}, & \text{если } x\le 0 \\ x, & \text{если } x>0 \end{cases} \qquad (5.3)$$

Алгоритмы, в которых в зависимости от выполнения некоторого условия или значения некоторого признака обработка информации производится по одному из нескольких возможных направлений, называются р а з в е т в л я ю щ и м и с я .

Каждое из возможных направлений обработки называется в е т в ь ю алгоритма. Обычно условие разветвления задается в виде операции отношения, использующей знаки :

$$=,\ <,\ \le,\ >,\ \ge,\ \neq .$$

5.3. Блок-схемы

В простых случаях алгоритм может быть сразу сформулирован в виде последовательности к о м а н д и с п о л н и т е л я , т.е. в виде, когда каждое предписание алгоритма п о н я т н о исполнителю и может быть им выполнено.

В более сложных случаях вопрос о детализации описания алгоритма решается постепенно, в несколько приемов. Сначала выделяют наиболее важные шаги с указанием их функций в довольно общей форме и устанавливают связи между ними. После того, как появится уверенность в правильности общей структуры алгоритма, производится его дальнейшая детализация путем разбиения крупных шагов на более мелкие и установления соответствующих связей между ними, пока, наконец, весь алгоритм не будет представлен в виде перечня команд исполнителя. Такой прием называют п о ш а г о в о й д е т а л и з а ц и е й или пошаговым уточнением алгоритма.

В любом случае алгоритм должен быть записан в компактной, наглядной и легко понимаемой форме. Одним из средств достижения этого требования является применение для описания алгоритмов б л о к - с х е м , или просто с х е м .

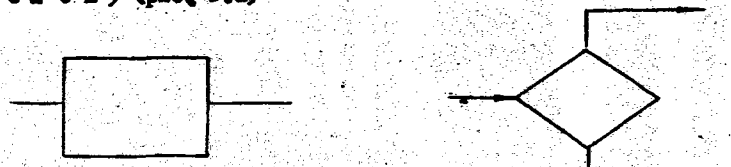
Б л о к - с х е м а - это наглядный графический способ представления алгоритмов. Отдельные предписания алгоритма

изображаются в форме определенных плоских геометрических фигур, называемых *символами* блок-схем, или *блоками*. Переходы от предписания к предписанию изображаются с помощью линий потока, а направления переходов – стрелками. Внутри блоков приводится описание содержания соответствующего шага алгоритмического процесса и правило выбора одного из нескольких возможных направлений алгоритма.

Условные графические обозначения (символы) и правила выполнения схем алгоритмов должны соответствовать требованиям ГОСТ 19.002-80 и ГОСТ 19.003-80. В таблице 5.2 приведены наименование, обозначение и функции основных обязательных символов. Направления линии потока сверху вниз и слева направо принимаются за основные, и, если линии потока не имеют изломов и пересечений, стрелками их можно не обозначать. Формулировка пояснений внутри символов схемы алгоритма относительно произвольна. Главное – четкое, ясное и точное описание последовательности шагов, которую необходимо соблюдать при исполнении алгоритма. В блок-схеме не должно быть ни недостижимых блоков, ни бесконечно повторяющихся участков, иначе она считается неправильной.

Для вычерчивания символов блок-схем используют специальный шаблон. При вычерчивании их вручную рекомендуется принимать длину символа приблизительно равной 1,5 его высоты.

Пожалуй, самыми существенными элементами блок-схем являются два блока: *функциональный*[1] *блок (арифметический, или блок присваивания)* и *блок принятия решения (логический блок)* (рис. 5.2)



а) функциональный (арифметический) блок

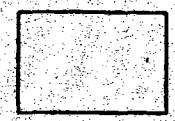
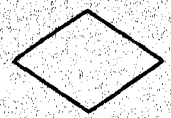
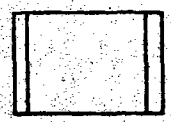
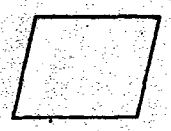
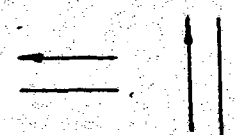
б) логический блок

Рис. 5.2

1) Термин используется в случае невычислительных алгоритмов.

Таблица 5.2

Перечень обозначений и наименований символов

Наименование	Обозначение (ГОСТ 19.003-80)	Функции
Процесс		Выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположение данных.
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий.
Предопределенный процесс		Использование ранее созданных и отдельно описанных алгоритмов или программ.
Ввод – вывод		Преобразование данных в форму, пригодную для обработки (ввод) или регистрация результата обработки (вывод).
Линии потока		Направление обработки данных, связь между символами блок-схемы.
Пуск-останов		Начало, конец, прерывание процесса обработки данных.

В невычислительных алгоритмах функциональный блок (рис.5.2) определяет некоторую последовательность действий (одна или несколько команд) исполнителя, связанных с последовательной обработкой информации и не требующих принятия каких-либо решений, например : "установить раствор циркуля равным длине отрезка AB ", "провести окружность", "поставить ножку циркуля в точку B " и т.д. Блок имеет один вход и один выход.

В алгоритмах работы с числовыми величинами этот блок задает чаще всего арифметические вычисления над входными и промежуточными величинами, и присваивание вычисленных значений промежуточным и результирующим (выходным) величинам алгоритма. Основная операция этого блока – п р и с в а и в а н и е , которую обозначают символами $:=$. Общий вид оператора (команды) присваивания :

$$V := \text{выражение}$$

Последняя запись означает, что по формуле, задающей "выражение", должны быть произведены вычисления при т е к у щ и х значениях входящих в нее переменных и полученный результат надо заменить п р е д ы д у щ е е значение переменной, стоящей слева от знака $:=$. В частности, переменная V может входить в "выражение", стоящее справа от знака $:=$. Например, $D := b^2 - 4ac$ при значениях a, b и c , соответственно равных 4, I, 2 обеспечивает вычисление величины $I - 4 \cdot 4 \cdot 2 = -3I$, которая в последующем становится значением переменной D и будет сохранять это значение до н о в о г о присваивания переменной D некоторого значения.

Иногда операцию присваивания обозначают стрелкой :

$$D \leftarrow b^2 - 4ac$$

Значение "выражения" можно считать определенным, когда всем входящим в него переменным ранее присвоены некоторые значения.

Может показаться, что мы обсуждаем совершенно очевидные вещи. Однако, у лиц, не искушенных в алгоритмизации и программировании, нередко вызывает недоумение операция $i := i + 1$, бессмысленная с точки зрения алгебры (если знак $:=$ рассматривать как эквивалентность !), но имеющая глубочайший смысл в программировании и означающая " i заменить на $i + 1$ ", т.е. к значению переменной i , которое она имела к началу выполнения операции

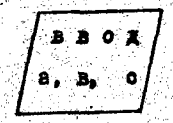
присваивания, прибавить число I и считать полученное значение н о в ы м значением переменной i . Прежнее значение i пропадает.

Обращаясь к ранее рассмотренному алгоритму Евклида, запишем его третий шаг так : $m := n$; $n := r$.

Эта операция совершенно отличается от такой: $n := r$; $m := n$.

В последнем случае предыдущее значение величины n было бы безнадежно утрачено прежде, чем его можно использовать для замещения m .

Блок в в о д а – в ы в о д а позволяет указывать, какие величины получили конкретные значения при вводе информации или какие данные выводятся для контроля и анализа промежуточных и окончательных результатов. Например, обозначение в начале блок-схемы



следует понимать так : "ввести извне (прочитать) три числа и присвоить первое из них переменной a , второе – переменной b , третье – переменной c ".

Из структуры алгоритма обычно ясно, относится ли блок, изображаемый параллелограммом, ко вводу или к выводу информации. В противном случае следует в блоке писать соответственно "читать (ввод)" или "записать (вывод)". Можно также для обозначения ввода и вывода информации использовать соответствующие символы ГОСТ 19.003-80, изображенные ниже.

а) перфокарта

б) документ

Блоков ввода-вывода и арифметического блока обычно оказывается достаточно для изображения линейных алгоритмов.

Пример 5.I. Составить схему алгоритма перехода к градусным мерам Фаренгейта и Реомюра, если измеряемая температура выражена

в градусах Цельсия.

Напомним, что переход от градусов Цельсия к градусам Фаренгейта (F) и Реомюра (R) осуществляется по формулам:

$$F = 1.8\,C + 32 \;; \quad R = 0.8\,C \;,$$

где C - градусная мера Цельсия.

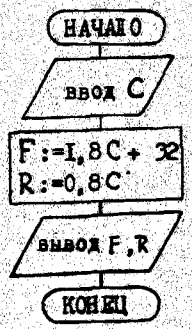


Рис. 5.3

Как видим, линейные алгоритмы наиболее просты по своей структуре и содержат лишь команды ввода исходных данных, команды непосредственных вычислений и команды вывода результатов.

Логический блок (рис. 5.2,б) обеспечивает проверку выполнения некоторого условия P и выбор, в зависимости от истинности или ложности проверяемого условия, одного из нескольких возможных путей продолжения алгоритма. Проверяемое условие задается обычно внутри ромба. Выбираемые пути могут помечаться метками " да/нет","+/-". Мы остановимся на последнем обозначении : если проверяемое условие выполняется (истинно), то происходит переход по стрелке "+" , если не выполняется (ложно) - по стрелке "-" .

Каждый из путей - S_1 или S_2 (рис. 5.4) - ведет к общей точке слияния B , так что дальнейшая обработка продолжается независимо от того, какой путь бы выбран.

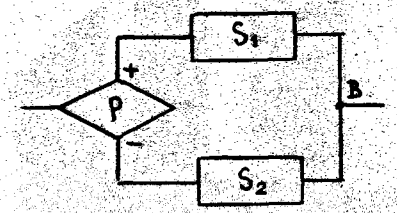


Рис. 5.4

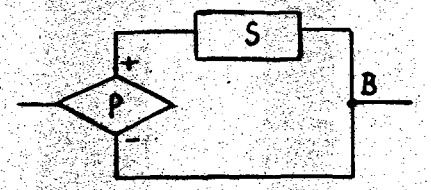


Рис. 5.5

Может оказаться, что для одного из результатов проверки условия P никаких действий предпринимать не надо. В этом случае рекомендуется оформлять фрагмент схемы, как показано на рис. 5.5.

Пример 5.2. Требуется определить, попадает ли точка с произвольно введенными координатами (X , Y) в кольцо, образуемое двумя концентрическими окружностями радиусов $r = 2$ и $R = 4$ и центрами в начале координат (рис. 5.6)

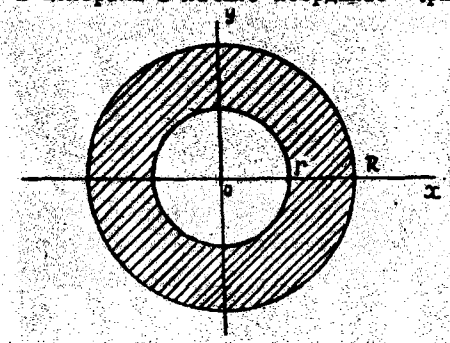


Рис. 5.6

Уравнение малой и большой окружностей имеют вид:

$$x^2 + y^2 = r^2$$
$$x^2 + y^2 = R^2$$

Очевидно, что точка с произвольно взятыми координатами X и Y попадает в область кольца, если одновременно выполняются неравенства

$$X^2 + Y^2 \geq r^2$$
$$X^2 + Y^2 \leq R^2$$

Другими словами, точка будет в кольце, если расстояние точки (X , Y) от начала координат $\rho = \sqrt{X^2 + Y^2}$ удовлетворяет неравенству $r \leq \rho \leq R$.

Схема алгоритма приведена на рис. 5.7. В блоке, обозначенном "ввод X , Y " , определяются два числа X и Y , вводимые

извне как координаты исследуемой точки плоскости xOy. В следующем блоке переменным R, r и ρ присваиваются соответствующие числовые значения, так что в последующих логических блоках сравниваются между собой вполне конкретные, ранее определенные величины: если $\rho < r$, то точка (X, Y) находится внутри малого круга и, стало быть, не попадает в кольцо; то же самое, если $\rho > R$ (второй логический блок). И только, если $\rho \geq r$, но $\rho \leq R$ – точка будет в кольце.

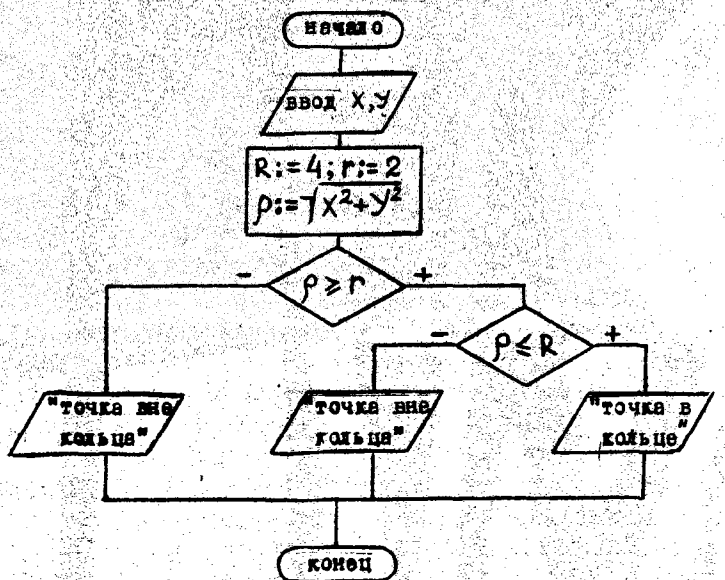


Рис. 5.7

В форме схем могут быть наглядно изображены многие алгоритмы, возникающие при изучении школьных предметов, причем, такое представление часто позволяет серьезно облегчить не только задачу ученика в освоении нового материала, но и учителя при объяснении и контроле знаний.

Пример 5.3. Рассмотрим процесс решения неравенства $ax < b$, где a, b – произвольные числа.

Если $a = 0$, то неравенство превратится в $0 \cdot x < b$ и теперь уже надо анализировать знак величины b. При $b > 0$ решением может быть любое число, иначе – решений нет.

Если же $a \neq 0$, то дополнительно необходимо исследовать его знак:

при $a > 0$ решение – любое $x < b/a$,

при $a < 0$ – $x > b/a$.

Схема алгоритма полного исследования и решения неравенства $ax < b$ приведена на рис. 5.8

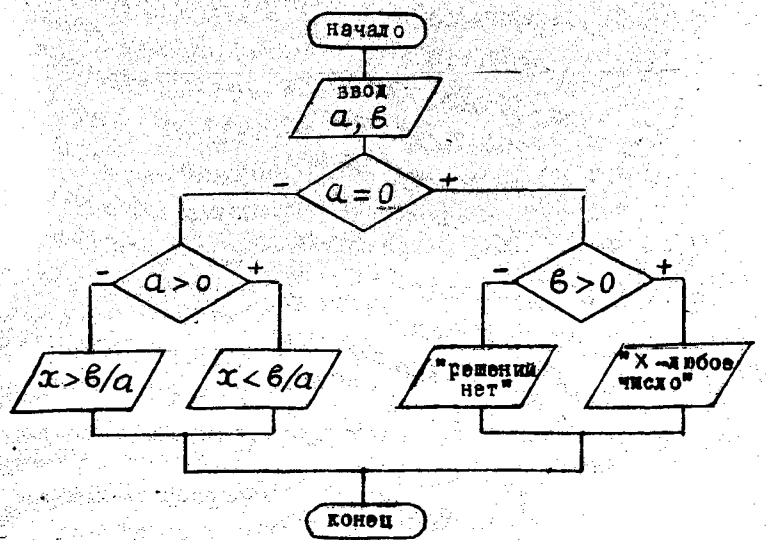


Рис. 5.8

Упражнения:

I. Составить схему алгоритма:

а) вычисления корней квадратного уравнения $ax^2 + bx + c = 0$ для произвольных действительных значений a, b, c.

б) упорядочения трех чисел x, y и z в порядке возрастания их значений.

в) определения принадлежности точки (X, Y) прямоугольнику (рис.5.9), если a, b, c, d известны.

г) определения принадлежности точки (X, Y) треугольнику (рис.5.10), заданному системой неравенств:

$$\begin{cases} y \geq 0 \\ y \leq a - |x| \end{cases}$$

д) определения, могут ли три, наугад взятые, числа быть длинами сторон треугольника.

2. Дан выпуклый четырехугольник $ABCD$ со сторонами $AB = a$, $BC = b$, $CD = c$, $DA = d$. Составить алгоритм, определяющий, можно ли в этот четырехугольник вписать окружность.

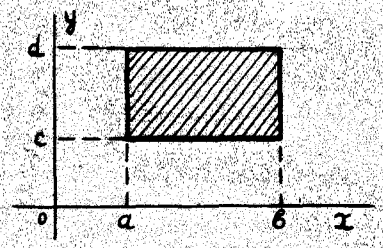


Рис. 5.9

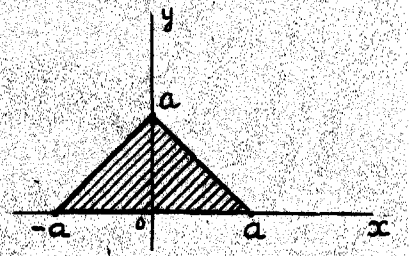


Рис. 5.10

5.4. Построчная запись алгоритмов. Понятие об алгоритмическом языке

Основное преимущество блок-схем заключается в том, что с их помощью можно наглядно изобразить структуру алгоритма в целом.

Обычно построение схемы алгоритма начинается с наиболее крупных, хорошо известных этапов обработки информации. Затем, в процессе обдумывания необходимых переходов, разветвлений, дополнительных проверок, основная схема уточняется, дополняется второстепенными блоками. Очевидно, что изображение алгоритма с большой степенью детализации в виде схемы будет слишком громоздким и потеряет всякую наглядность. Так достоинства блок-схем постепенно переходят в их недостатки. К недостаткам блок-схем следует отнести также трудности автоматического воспроизведения их на бумаге или экране дисплея.

Наиболее удобным и широко распространенным способом описания алгоритмов является так называемая **построчная алгоритмическая запись** (нотация) с помощью специального **алгоритмического языка**. Иногда такая форма записи называется **псевдокодом**.

Обычно в литературе по информатике под термином "алгоритмический язык" понимают язык, пригодный для точного и полного описания алгоритма, так и для непосредственного составления программы для ЭВМ.

Наиболее популярными языками являются ФОРТРАН, АЛГОЛ, КОБОЛ, ПЛ/I, БЭЙСИК.

ФОРТРАН относится к наиболее ранним языкам программирования. Он был разработан в 1955-56 гг. специалистами всемирно известной фирмы IBM (США), специализирующейся по производству ЭВМ и их программного обеспечения. Этот язык ориентирован на реализацию алгоритмов, в которых преобладают формулы. Широко используется при решении инженерных и научных задач, но может с успехом применяться и при решении экономических и информационно-логических задач. Язык оказался очень удобным в использовании. Благодаря простоте и ясности его конструкций машинные программы с этого языка оказываются обычно более эффективными, чем программы, полученные с других языков (машинная программа получается после обработки программы на исходном языке - ФОРТРАН, ПЛ/I и т.д. - специальной программой, называемой **транслятором**).

АЛГОЛ-60 сохраняет свое значение в качестве языка для записи алгоритмов, не утратил свое значение как язык непосредственного программирования для ЭВМ.

И ФОРТРАН и АЛГОЛ продолжают развиваться. Разработаны языки АЛГОЛ-68 и ФОРТРАН-77 - более совершенные и удовлетворяющие современным требованиям технологии обработки данных на ЭВМ.

ФОРТРАН послужил основой диалогового языка БЭЙСИК, который успешно применяется для обучения навыкам программирования.

КОБОЛ - ориентирован на решение коммерческих задач. Имеет хорошо развитые средства для описания данных сложной структуры. К недостаткам относится его многословность (программа на КОБОЛЕ состоит из полных слов английского или русского языка типа ВЫЧИСЛИТЬ, ПОВТОРИТЬ и др.) и слабо развитые процедурные средства обработки данных.

ПЛ/I - разработанный в период 1963-66 гг. сотрудниками IBM, вобрал в себя достоинства предшествующих языков программиро-

вания ФОРТРАН, КОБОЛ, АЛГОЛ и некоторых других, в частности, языков для обработки текстовой информации.

В то же время язык ПЛ/I содержит новые средства, обеспечивающие использование разного типа данных (в том числе работу с двоичной информацией), создание и обработку сложных структур данных, возможность программной обработки прерываний вычислительного процесса, организацию параллельных вычислений, использование большого числа стандартных процедур.

Язык завоевал широкое признание; успешно применяется для решения не только инженерных, научно-технических и экономических задач, но и всевозможных информационно-логических задач, создания поисковых систем, задач анализа и изучения произведений литературы, музыки и искусства.

Кроме упоминавшихся языков ФОРТРАН-77 и АЛГОЛ-68, в ближайшее время широкое распространение, очевидно, найдут такие языки, как ПАСКАЛЬ и АДА, наиболее полно учитывающие современные тенденции и технологию программирования.

Для упомянутых выше языков сохраним общее название "языки программирования высокого уровня", или просто "языки программирования".

Под термином же "алгоритмический язык" мы будем понимать с и с т е м у о б о з н а ч е н и й и п р а в и л для е д и н о о б р а з н о й и т о ч н о й з а п и с и а л г о р и т м о в в словесной форме.

Этот язык близок к естественному языку, но достаточно формализован, чтобы удовлетворять основным требованиям, предъявляемым к алгоритмам: определенности, массовости, результативности и др.

Алгоритмический язык, кстати, как и язык блок-схем, включает в себя без ограничений математическую символику: числа, обозначения величин и функций, знаки операций, скобки и др.

При записи на алгоритмическом языке различных невычислительных алгоритмов команды исполнителя обычно выглядят как повелительные предложения русского языка. Запись вычислительных алгоритмов более формализована и так как этот язык ориентирован на исполнителя-человека, будем считать, что в системе команд такого исполнителя возможно выполнение "за один шаг", например, таких вычислений :

$$y := \sqrt{x} \quad ; \quad b := arctg\, z$$

Таким образом, от обычной словесной записи алгоритмов, применявшейся в курсе "Алгебра-8" и нами в §1, алгоритмическая запись отличается большей формализацией, точностью и четкостью конструкций.

В алгоритмическом языке используется некоторое ограниченное число слов, смысл и способ употребления которых заданы раз и навсегда. Эти слова называются с л у ж е б н ы м и с л о в а м и. При записи алгоритмов они подчеркиваются и относятся к неделимым символам. Подчеркивание служебных слов делает запись алгоритмов более наглядной и понятной. Введем эти служебные слова :

ЧИТАТЬ, ЗАПИСАТЬ, ЕСЛИ, ИДТИ К, КОНЕЦ, АЛГОРИТМ.

Основными объектами в алгоритмической записи являются о п е р а т о р ы. Наиболее распространенным является ф у н к ц и о н а л ь н ы й о п е р а т о р (см. п.3) или оператор общей обработки, предписывающий некоторые конкретные действия, связанные с изменением значения, формы представления или расположения информации. Пример : вынуть из урны один шар; проверить наличие патрона в патроннике. Частным и наиболее важным в алгоритмизации случаем этого оператора является оператор присваивания (см. п.3) :

V := выражение

Примеры записи операторов присваивания

$$\pi := 4\left(4\,arctg\,\frac{1}{5} - arctg\,\frac{1}{239}\right) \qquad \text{(формула Мачина)}$$

$$H := H_0 + V_0 t + gt^2/2$$

Алфавит рассматриваемого учебного алгоритмического языка специально ничем не ограничен; в нем могут быть использованы любые понятные школьнику символы: строчные и прописные буквы русского, латинского и греческого алфавитов, знаки арифметических операций + , - , x, / , а также знаки операций отношения = , ≠ , < , > , ≤ , ≥ и скобки.

На форму записи в ы р а ж е н и й не накладывается никаких ограничений - здесь можно использовать любые математические знаки, подстрочные и надстрочные индексы.

Л о г и ч е с к и е в ы р а ж е н и я представляются опе-

равными отношения. О т н о ш е н и е состоит обычно из двух арифметических выражений, соединенных знаком операции отношения, например, $x^2+y^2<\rho$, $a \neq 6$ и т.д.

В вычислительных алгоритмах широко используется понятие в е л и ч и н ы. Величины делятся на постоянные и переменные. Постоянной называется величина, значение которой не меняется в процессе выполнения алгоритма. Переменной называется величина, значение которой может изменяться в процессе выполнения алгоритма. При записи алгоритма для обозначения переменных величин вводятся имена (идентификаторы) аналогично обозначениям переменных в школьном курсе алгебры. Такое обозначение переменной величины в алгоритмическом языке называется и м е н е м в е л и ч и н ы.

При исполнении алгоритма в каждый момент времени величина обычно имеет некоторое значение, которое называется ее т е к у щ и м значением. Если величина используется в вычислениях, не получив в предшествующих операторах некоторого значения, то она называется н е о п р е д е л е н н о й.

Будем различать в алгоритме входные величины (а р г у м е н т ы), п р о м е ж у т о ч н ы е величины и выходные величины (р е з у л ь т а т ы).

Кроме числовых величин, в построчной записи алгоритмов будем использовать также т е к с т о в ы е или л и т е р н ы е величины. Чаще всего они будут использоваться в операторах записи данных.

Операторы чтения исходных данных и записи результатов имеют вид :

читать $a, 6, \ldots, z$
записать $x, a, y \ldots, d$

Предполагается, что в результате выполнения первого оператора всем величинам, перечисленным в нем через запятую, присваиваются числовые значения, заранее приготовленные для этого конкретного исполнения алгоритма. Точно такое исполнение оператора з а п и с а т ь означает фиксирование каким-либо способом (например, записью на бумагу) тех конкретных числовых значений перечисленных в операторе величин, которые эти величины имеют в момент выполнения оператора.

В операторе записать в качестве элемента списка нередко используются упоминавшиеся выше текстовые величины, заключаемые в кавычки. Так, оператор

записать "сумма $S=$", S

предписывает исполнителю зафиксировать, скажем, на бумаге, текст с у м м а S = (уже без кавычек), вслед за которым выводится текущее числовое значение величины S , например :

с у м м а S = 175.3

Запись алгоритма на алгоритмическом языке представляет собой последовательность п р о н у м е р о в а н н ы х строк. Строки нумеруются, начиная с единицы; номер от оператора отделяется точкой. В конце строки никакой знак не ставится. В каждой строке может размещаться один или несколько операторов. В последнем случае операторы отделяются друг от друга точкой с запятой, например:

5. $D:=6^2-4ac$; $e_1:=-6/2a$; $e_2:=\sqrt{D}/2a$

Операторы в строке выполняются последовательно слева направо. Если оператор не помещается в одну строку, он может быть продолжен на следующей, однако, эта новая строка не нумеруется.

Для единообразного оформления и присвоения алгоритму имени он снабжается з а г о л о в к о м , с которого, собственно, и начинается алгоритм. Заголовок имеет вид :

Алгоритм И М Я

Здесь Алгоритм - служебное слово, ИМЯ - краткое название алгоритма, отражающее его содержание. Рекомендуется имя записывать заглавными буквами, например :

Алгоритм Н О Д

Пример 5.4. Приведем построчную запись алгоритма для решения следующей задачи.

В трех сосудах содержится вода. В первом сосуде содержится V_1 л воды температуры t_1 , во втором - V_2 л температуры t_2, в третьем - V_3 л температуры t_3 .

Найдите объем и температуру воды после слияния содержимого трех этих сосудов в один сосуд (изменением объема воды при изменении температуры пренебречь).

Обозначим объем воды и ее температуру после слияния соответственно через V и t. Тогда с учетом принятых в алгоритмическом языке соглашений построчная запись алгоритма будет иметь вид.

алгоритм СМЕСЬ

1. **читать** $V_1, t_1, V_2, t_2, V_3, t_3$
2. $V := V_1 + V_2 + V_3$
3. $r := V_1 t_1 + V_2 t_2 + V_3 t_3$
4. $t := r/V$
5. **Записать** "объем =" , V , "температура =", t
6. **конец**

Несмотря на то, что в приведенной построчной записи алгоритма ничего не говорится о порядке выполнения операторов, по умолчанию здесь и далее везде предполагается, что операторы алгоритма выполняются в порядке их естественного расположения; другими словами - в порядке возрастания номеров строк алгоритма.

Такой порядок выполнения предписаний, однако, может быть нарушен с помощью операторов **управления**. К операторам управления относятся: оператор безусловного перехода, оператор условного перехода и оператор остановки.

Оператор **безусловного перехода** записывается так:

идти к НС

где **идти к** - служебное слово, а НС - номер строки алгоритма в построчной нотации.

После выполнения оператора безусловного перехода следующим выполняется оператор, находящийся в строке с номером НС. Например,

8. **идти к** 4

означает безусловный переход к ранее выполнявшемуся оператору в строке с номером 4.

Оператор безусловного перехода не обязательно занимает отдельную строку алгоритма; он может располагаться в строке вслед за другими операторами, но так, чтобы других операторов после него в строке не было.

Оператор **условного перехода** имеет вид:

если Р **идти к** НС

где **если**, **идти к** - служебные слова, Р - проверяемое условие, НС - номер строки оператора, которому передается управление, если условие Р выполняется; в противном случае выполняется оператор, находящийся первым в строке, следующей непосредственно за оператором безусловного перехода.

Оператор условного перехода соответствует логическому блоку блок-схемного представления алгоритмов и служит для организации разветвлений в алгоритмах, представленных в построчной алгоритмической нотации.

Пример записи оператора:

5. **если** $b^2 - 4ac < 0$ **идти к** 13
6. .....

Здесь выражение $b^2 - 4ac$ вычисляется при текущих значениях входящих в него величин a, b, c и полученное числовое значение сравнивается с нулем. Если условие истинно, то следующим выполняется оператор в строке с номером 13; в противном случае - оператор в 6-й строке.

Для завершения выполнения алгоритма служит оператор **остановки**, который записывается в виде служебного слова **конец**.

Практическая реализация всех предусмотренных алгоритмом действий по получению результата на основе подготовленных конкретных значений аргументов осуществляется в процессе исполнения алгоритма.

Для лучшего понимания алгоритма полезно иногда выполнять так называемую "ручную прокрутку" или "проигрывание" алгоритмической записи. Для этого составляется таблица, содержащая колонки с именами имеющихся в алгоритме входных величин (аргументов), промежуточных величин и выходных величин (результатов). В таблицу вписываются числовые значения всех величин, которые те получают в процессе выполнения алгоритма. Первая графа содержит номера предписаний, а последняя служит для проверки условий и некоторых пояснений.

Пример 5.5. Выполним "ручную прокрутку" алгоритма нахождения наибольшего из трех заданных чисел x, y, z.

а) блок-схема

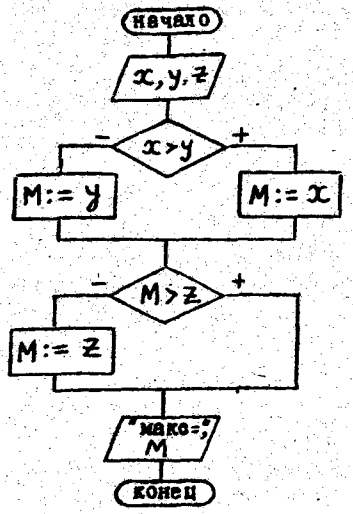


Рис. 5.II.

б) построчная алгоритмическая нотация

алгоритм Б И Т

1. читать x , y , z
2. если $x>y$ идти к 4
3. M := y ; идти к 5
4. M := x
5. если M > z идти к 7
6. M := z
7. записать "максимум", M
8. конец

Весь механизм происходящих в алгоритме преобразований аргументов в результат вскрывается в процессе "проигрывания" алгоритма (см. табл. 5.3).

Таблица 5.3

Шаги алгоритма	Аргументы			Результат	Проверка условий, пояснения
	X	y	z	M	
I.	I3	6	2I		чтение x, y, z
2.					I3>6 ? да
4.				I3	
5.					I3> 2I ? нет
6.				2I	
7.					максимум= 2I
8.					останов

Пусть в качестве исходных данных исполнитель получает три числа: I3, 6 2I . Эти данные заносятся на первом шаге алгоритма в таблицу в качестве аргументов x , y , z .

На втором шаге согласно алгоритму проверяется условие $x>y$. Текущими значениями переменных x и y , как видно из таблицы, являются числа I3 и 6 , поэтому проверяемое условие в данном случае соблюдается. Результат проверки условия для контроля заносится в соответствующую строку последней колонки таблицы: записывается условие и за ним слово "да", если условие выполняется, и "нет" в противном случае. Так как на втором шаге условие соблюдено, то надо переходить к шагу 4, что отражается в первой графе таблицы.

Четвертый шаг алгоритма - выполнение операции присваивания M := X, согласно которому исполнитель присваивает переменной M значение I3, которое и вносится в таблицу 5.3

На пятом шаге, к которому мы переходим в порядке естественного расположения предписаний алгоритма, проверяется условие M > z . Так как к этому моменту M = I3 (см. таблицу), а z равно 2I, то в результате проверяемое условие не удовлетворяется, вследствие чего исполнитель переходит к предписанию, в котором выполняется присваивание M : = z и M , таким образом, получит значение 2I . Предыдущее значение M при повторном присваивании теряется и для наглядности в таблице перечеркивается. Переменная M здесь играет двоякую роль: она используется как промежуточная переменная в ходе выполнения алгоритма, а на выходе содержит результат и в таблице отмечена как результирующая переменная.
В общем случае промежуточные переменные также вносятся в таблицу под соответствующим заголовком.

Пример 5.6. Алгоритм решения квадратного уравнения $ax^2 + bx + c = 0 \ (a \neq 0)$ в области действительных чисел.

а) блок-схема

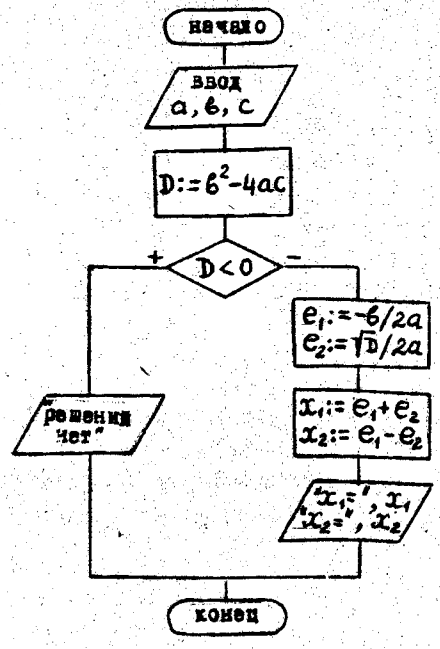


Рис. 5.12

б) построчная алгоритмическая запись

Алгоритм КВУР

1. читать a, b, c
2. $D := b^2 - 4ac$
3. если $D < 0$, идти к 8
4. $e_1 := -b/2a$;
   $e_2 := \sqrt{D}/2a$
5. $x_1 := e_1 + e_2$;
   $x_2 := e_1 - e_2$
6. записать "$x_1=$", x_1, "$x_2=$", x_2
7. идти к 9
8. записать "решений нет"
9. конец.

Пример 5.7. Составим блок-схему и построчную запись алгоритма вычисления значения функции y, заданной графически (5.13). Очевидно, что

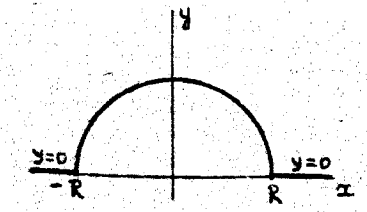


Рис. 5.13

$$y = \begin{cases} 0, & \text{если } x \leq -R \\ \sqrt{R^2 - x^2}, & \text{если } -R < x < R \\ 0, & \text{если } x \geq R \end{cases}$$

а) блок-схема

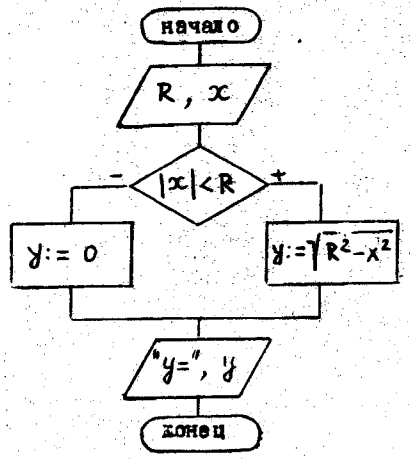


Рис. 5.14

б) построчная алгоритмическая запись

Алгоритм ГРАФ

1. Читать R, x
2. если $|x| < R$, идти к 4
3. $y := 0$; идти к 5
4. $y := \sqrt{R^2 - x^2}$
5. записать "$y=$", y
6. конец.

Пример 5.8. На плоскости в прямоугольной системе координат заданы треугольник координатами своих вершин:

$$A(x_1, y_1), \ B(x_2, y_2), \ C(x_3, y_3)$$

Составить алгоритм, определяющий, является ли данный треугольник равносторонним, равнобедренным или разносторонним

а) блок схема

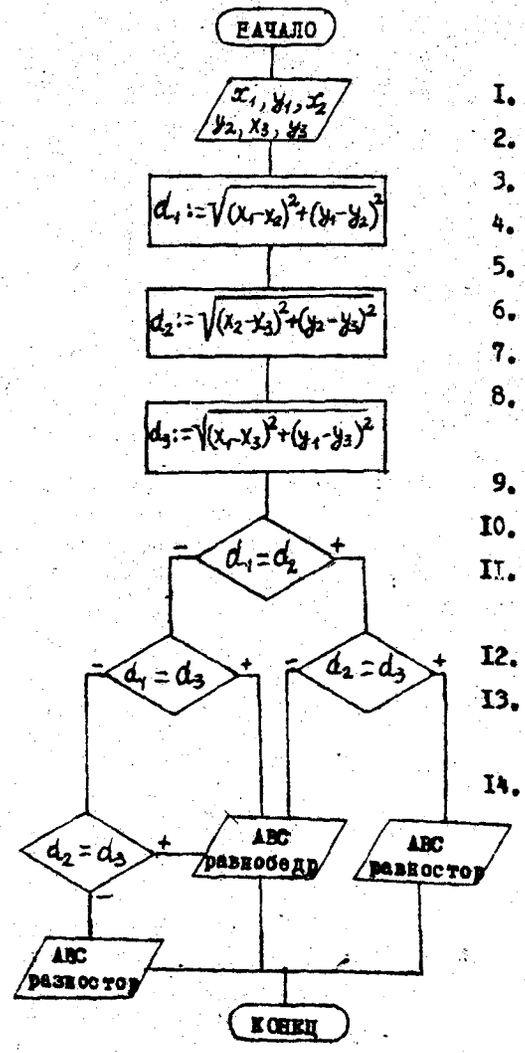


Рис. 5.15

б) построчная алгоритмическая запись

Алгоритм треугольник

I. читать $x_1, x_2, x_3, y_1, y_2, y_3$
2. $d_1 := \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$
3. $d_2 := \sqrt{(x_2-x_3)^2+(y_2-y_3)^2}$
4. $d_3 := \sqrt{(x_1-x_3)+(y_1-y_3)^2}$
5. если $d_1 = d_2$ идти к I0
6. если $d_1 = d_3$ идти к I3
7. если $d_2 = d_3$ идти к I3
8. записать "треугольник разносторонний"
9. идти к I4
I0. если $d_2 \neq d_3$ идти к I3
II. записать "треугольник равносторонний"
I2. идти к I4
I3. записать "треугольник равнобедренный"
I4. конец.

Упражнения

I. Составить блок-схемы и построчную запись алгоритмов следующих задач :

а) решить неравенство $ax > 6$

б) вычислить абсолютную величину числа x .

У к а з а н и е . Воспользоваться определением абсолютной величины.

$$|x| = \begin{cases} x, & \text{если } x \geq 0 \\ -x, & \text{если } x < 0 \end{cases}$$

2. Выполнить построчную запись алгоритмов всех задач из упражнения II5.3.

3. Составить алгоритмы решения следующих задач :

а) вычислить остаток от деления нацело двух целых положительных чисел m и n $(m \div n)$.

б) определить, какая из двух данных точек $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$ на плоскости в прямоугольной системе координат расположена ближе к началу координат.

§ 6. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

В большинстве случаев алгоритмы содержат участки, которые в ходе работы алгоритма повторяются многократно. Такие участки называются ц и к л а м и , а алгоритмы ц и к л и ч е с к и м и.

Пример 6.I. Пусть требуется вычислить и записать несколько значений функции

$$y = x^2 - x + 41$$

для $x \in N$ (N – множество натуральных чисел).

Выражение, стоящее справа – известный трехчлен Эйлера, значениями которого при $x = 1, 2, \ldots, 40$ будут простые числа.

Последовательность действий, необходимых для решения поставленной задачи, может быть записана так:

1. $x := 1$
2. $y := x^2 - x + 41$
3. записать x, y
4. $x := x + 1$
5. $y := x^2 - x + 41$
6. записать x, y
7. $x := x + 1$
8. $y := x^2 - x + 41$

и т.д.

Очевидно, что желая получить значения функции для последующих значений x, мы должны были бы и далее записывать абсолютно одинаковые тройки предписаний, таких, как предписания с номерами 2,3,4 или 5,6,7 и т.д. Эти последовательности, правда, можно не выписывать; важно уметь "заставить" многократно выполняться повторяющуюся последовательность:

$y := x^2 - x + 41$
записать x, y
$x := x + 1$

Используем для этой цели оператор безусловного перехода :

1. $x := 1$
2. $y := x^2 - x + 41$
3. записать x, y
4. $x := x + 1$
5. идти к 2

Действительно, в результате выполнения этой последовательности предписаний эффект получается тот же, что и в случае, когда тройки типа 2,3,4 были явно выписаны несколько раз подряд.

Наглядно динамика описанного процесса видна из блок-схемы (рис. 6.I а)

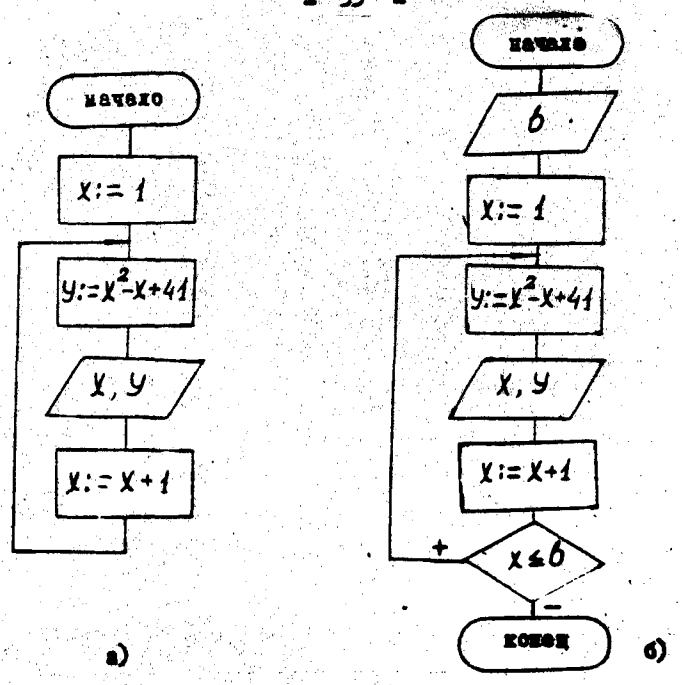


Рис. 6.I

Мы получили пример ц и к л и ч е с к о г о алгоритма. В процессе исполнения этого алгоритма нетрудно заметить, что в нем нарушено одно из условий, предъявляемых к алгоритмам – к о н е ч н о с т ь . Недостаток этот легко устранить, если учесть, что алгоритм дает верные результаты (генерирует простые числа) при $X \leq 40$. Дополним схему алгоритма логическим блоком, как показано на рис. 6.I,б., т.е. при каждом "обороте" цикла будем проверять условие $x < b$, где $b \leq 40$, вводится извне. Как только оно перестает выполняться, процесс заканчивается. Незначительными добавлениями алгоритм можно преобразовать так, что он будет применим на любом отрезке $[a, b]$ из области допустимых значений X и с любым $h \in N$, $h < 40$.

Окончательно алгоритм в построчной нотации выглядит так :

алгоритм ЭЙЛЕР

1. читать a, b, h
2. $x := a$
3. $y := x^2 - x + 41$
4. записать x, y
5. $x := x + h$
6. если $x \le b$ идти к 3
7. конец.

Лучше понять сущность циклических алгоритмов помогает процедура "ручной прокрутки". В таблице 6.I. показан последовательный процесс вычисления простых чисел по алгоритму ЭЙЛЕР при изменении X от I до 3 с шагом $h = I$ (вообще говоря, шаг может быть переменной величиной).

Содержимое первого столбца таблицы 6.I наглядно демонстрирует основное отличительное свойство циклических алгоритмов : количество исполняемых в процессе работы циклического алгоритма предписаний может существенно перекрывать количество предписаний, из которых составлена запись такого алгоритма.

Если количество повторений цикла известно, то для организации выхода из него производится подсчет количества повторений. Это делается с помощью вводимой промежуточной переменной, называемой с ч е т ч и к о м циклов. Выход из цикла происходит, как только счетчик превысит заданное количество.

Таблица 6.I

Шаги алгоритма	Аргументы				Результат	Проверка условий, пояснения
1.		I	3	I		читать a, b, h
2.	X					
3.					XX	
4.						записать I; 4I
5.	X					
6.						2 ≤ 3 ? да
3.					XX	
4.						записать 2 ; 43
5.	X					
6.						3 ≤ 3 ? да
3.					47	
4.						записать 3; 47
5.	4					
6.						4 ≤ 3 нет
7.						остановка

СТРУКТУРА ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Каждое очередное повторение цикла производится с новыми значениями обрабатываемых данных. Они появляются либо в результате предыдущего выполнения цикла, либо за счет выборки (чтения) новых обрабатываемых данных (в частности, за счет выборки элементов, содержащихся в таблицах - см. далее).

Назовем **параметрами цикла** переменные величины, которые используются для подготовки очередного повторения цикла (в предыдущем примере X - параметр цикла). К параметрам относятся аргументы вычисляемых функций, счетчики циклов и др. Во время выполнения цикла параметры меняются от некоторого начального до некоторого конечного значения. Поэтому для правильной организации циклического процесса надо установить начальные значения параметров, закон их изменения и условие окончания цикла. Иногда предписания алгоритма, выполняемые в цикле многократно, называют **телом** цикла.

Циклы, в которых только один вход и выход, нет разветвлений и других (вложенных) циклов, называются **простыми**, в противном случае - **сложными** циклами.

Структура простого цикла может быть изображена следующей схемой.

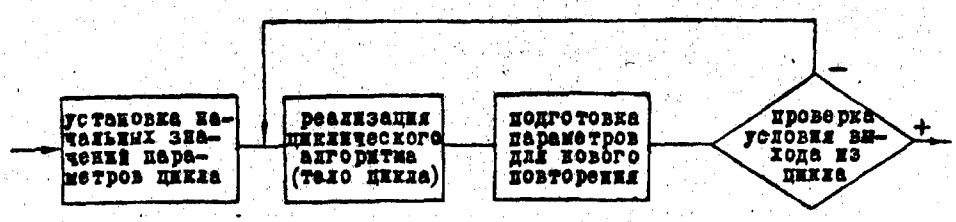


Рис. 6.2

Пример 6.2. Составить алгоритм вычисления суммы квадратов натуральных чисел от 1 до n.

Требуется найти $S = 1^2 + 2^2 + 3^2 + \dots n^2 = \sum_{i=1}^{n} i^2$

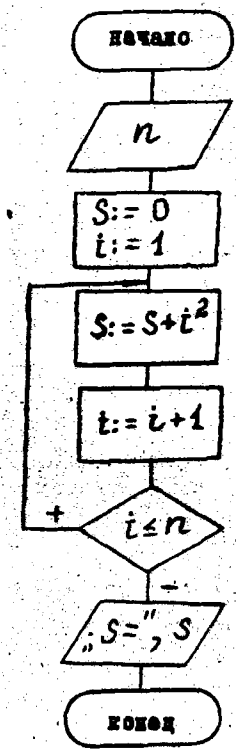
Для циклического накапливания сумм при составлении соответствующих алгоритмов используется схема

сумма := сумма + слагаемое

Если это предписание повторять требуемое количество раз, изменяя соответствующим образом "слагаемое", то и будет получена искомая сумма. Этот процесс напоминает суммирование на русских счетах.

Понятно, что "сумма" перед началом работы цикла должна иметь нулевое значение (или, как говорят, должна быть "очищена").

а) блок-схема



б) построчная алгоритмическая запись

алгоритм сумма

1. **читать** n
2. $S := 0$; $i = 1$
3. $S := S + i^2$
4. $i := i + 1$
5. **если** $i \le n$, **идти к** 3
6. **записать** „$S =$", S
7. **конец**

Рис. 6.3

На рис. 6.3. роль "суммы" играет переменная S, а роль "слагаемого" - квадрат очередного натурального числа. Изменение "слагаемого" достигается за счет последовательного увеличения счетчика цикла i на единицу. Если бы потребовалось вычислить, например, сумму квадратов **нечетных** натуральных чисел, то этот параметр следовало бы изменять по закону $i := i + 2$.

Полезно выполнить "ручную прокрутку" данного алгоритма. По такой же схеме можно разрабатывать алгоритмы для нахождения суммы заданного числа членов последовательности, определенной своим общим членом $a_k = f(k)$.

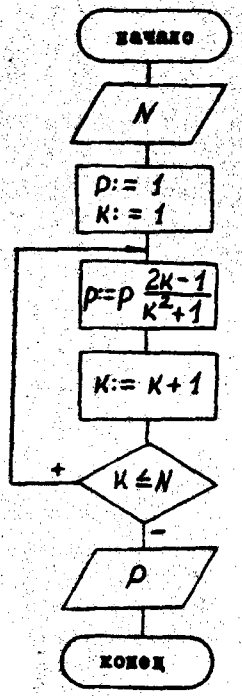
Пример 6.3. Составить алгоритм вычисления произведения первых членов последовательности с общим членом

$$a_k = \frac{2k-1}{k^2+1}$$

Для циклического получения последовательных значений произведений меняющихся множителей необходимо за пределом цикла положить вначале значение некоторой величины P – равной единице, а затем повторять требуемое количество раз предписание вида :

$P := P \times$ множитель

а) блок-схема



б) построчная алгоритмическая запись

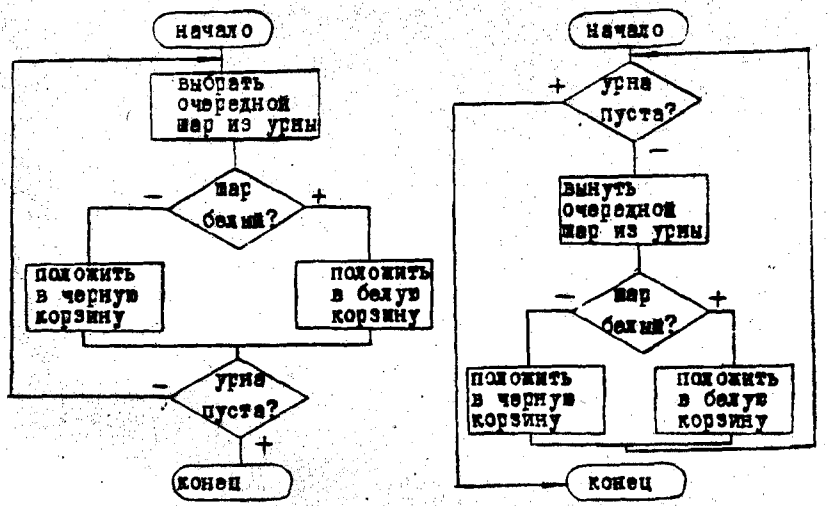
алгоритм ПРОИЗВЕДЕНИЕ
1. читать N
2. $P := 1$; $K := 1$
3. $P := P \cdot (2K-1)/(K^2+1)$
4. $K := K+1$
5. если $K \le N$ идти к 3
6. записать „$P =$", P
7. конец.

Рис. 6.4.

В виде циклических алгоритмов реализуются многие невычислительные задачи.

Пример 6.4. Составить алгоритм решения следующей задачи.

В урне находится некоторое количество черных и белых шаров. Требуется рассортировать их по двум корзинам – белой и черной – так, чтобы все черные шары оказались в черной корзине, а все белые – в белой.



а) цикл с постусловием б) цикл с предусловием

Рис. 6.5.

Последовательность действий, необходимых для решения поставленной задачи, хорошо видна из блок-схемы (рис. 6.5,а), которая составлена в предположении, что урна с самого начала не пуста; понятно, что в противном случае выполнить этот алгоритм было бы невозможно. Если такая ситуация не исключается, то правильным решением задачи будет схема, изображенная на рис.6.5,б.

Оба приведенные варианта различаются в алгоритмическом языке и называются соответственно цикл с постусловием и цикл с предусловием. Для первого характерно расположение тела цикла до проверки логического условия окончания цикла и как следствие – выполне-

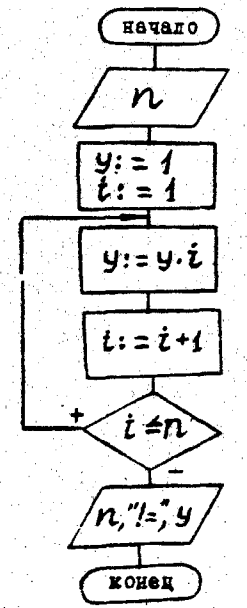
ние его хотя бы один раз, даже если проверяемое условие уже на первом "обороте" цикла не выполняется. В то же время тело цикла с предусловием в определенных условиях может не выполняться ни разу, ввиду расположения тела цикла после проверяемого условия.

Пример 6.5. Составить алгоритм вычисления $y = n!$ ($n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$). Считать, что значение n вводится извне.

Если положить вначале $y := 1$ и, изменяя параметр цикла i от I до n, выполнить n раз оператор $y := y \cdot i$, то значением переменной y как раз и будет $n!$

Алгоритм имеет вид :

а) блок-схема



б) построчная алгоритмическая запись

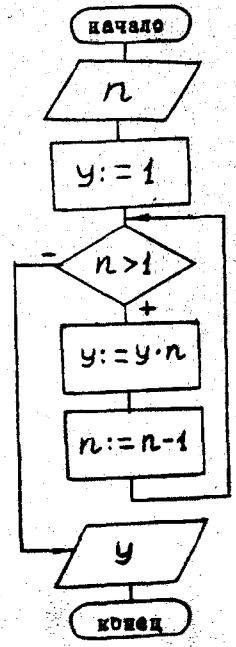
алгоритм ФАКТОРИАЛ

I. читать n
2. $y := 1$, $i = 1$
3. $y := y \cdot i$
4. $i := i + 1$
5. если $i \leq n$ идти к 3
6. записать n, "!=", y
7. конец.

Рис. 6.6

Этот алгоритм реализован как цикл с постусловием. Запишем его теперь как цикл с предусловием; кроме того, откажемся от вспомогательной переменной - счетчика цикла i и будем проводить вычисление $n!$ в таком порядке :

а) блок-схема



б) построчная алгоритмическая запись

алгоритм ОБРАТНА

I. читать n
2. $y := 1$
3. если $n \leq 1$ идти к 6
4. $y := y \cdot n$
5. $n := n - 1$ идти к 3
6. записать y
7. конец

Рис. 6.7

В обоих рассмотренных вариантах вычисления $n!$ предполагается, что $n \geq 0$ (напомним, что по определению $0! = 1$). Недостаток последнего варианта - не сохраняется исходное значение n. (Проверьте самостоятельно, при каком значении n оба алгоритма дают неверные результаты).

В рассмотренных выше алгоритмах условие окончания цикла либо определялось по явно заданному количеству повторений (по счетчику циклов), либо выход из цикла осуществлялся при превышении параметром цикла своего конечного значения.

Во многих циклических процессах число шагов, нужных для решения задачи, заранее определить либо невозможно, либо очень трудно.

Алгоритмы решения многих математических задач, в особенности тех, для которых не удается получить ответ в виде готовой формулы, основаны на следующей вычислительной процедуре: строится бесконечный процесс, сходящийся к искомому решению.

Он обрывается на некотором шаге (вычисления нельзя продолжать бесконечно) и полученная, таким образом, величина приближенно принимается за решение рассматриваемой задачи. Сходимость процесса гарантирует, что для любой наперед заданной точности $\varepsilon > 0$ найдется такой номер шага n, что на этом шаге ошибка в определении решения задачи не превысит ε.

Рассмотрим, например, как можно с заданной точностью вычислить число π посредством ряда:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^n \frac{1}{2n+1} \qquad (6.1)$$

Члены ряда образуют бесконечную убывающую последовательность. Каждый очередной член ряда получается из предыдущего с помощью следующих операций:

а) знаменатель предыдущего члена увеличивается на 2;
б) знак предыдущего члена меняется на противоположный.

Алгоритм сводится к последовательному циклическому вычислению и суммированию каждого очередного члена ряда, начиная с первого, равного 1; процесс продолжается, пока не будет найден член ряда, не превосходящий по абсолютной величине заданное число $\varepsilon > 0$. В математике доказано, что ошибка при этом не превосходит первого отброшенного члена ряда.

Такие циклические алгоритмы, в которых заранее неизвестно число повторений и проверка условия их окончания происходит по достижении нужной т о ч н о с т и, называют и т е р а ц и о н н ы м и (итерация – результат неоднократного применения какой-нибудь вычислительной процедуры).

Составим алгоритм поставленной выше задачи (рис.6.8).

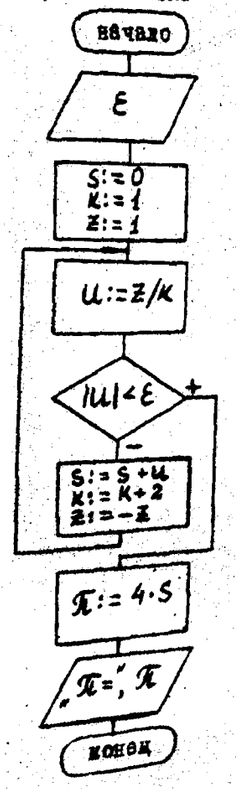
Параметр цикла K меняется с шагом 2 ($K = K + 2$), обеспечивая правильное формирование знаменателя очередного слагаемого. Вспомогательная переменная Z введена для очередного изменения знака перед слагаемым, благодаря чему оператор $U := \frac{Z}{K}$ в цикле создает последовательность

$$1, -\frac{1}{3}, \frac{1}{5}, -\frac{1}{7}, \frac{1}{9}, \ldots$$

$S := S + U$ формирует требуемую для вычисления

...ание для создания знакочередую-
...чаях нерационально.

а) блок-схема



б) построчная запись алгоритма

```
алгоритм ^ПИ
I. читать ε
2. S:=0; K:=1;
3. Z:=1
4. U:=Z/K
5. если |U|<ε идти к I0
6. S:=S+U
7. K:=K+2
8. Z:=-Z;
9. идти к 4
I0. π:=4·S
II. записать „π=", π
I2. конец.
```

Рис. 6.8

Пример 6.6. Составить алгоритм решения следующей задачи. Найти $y = \sqrt{X}$ с точностью $\varepsilon > 0$, пользуясь итерационной формулой Ньютона:

$$y_{n+1} = \frac{1}{2}\left(\frac{X}{y_n} + y_n\right), \quad n = 0, 1, 2, \ldots \qquad (6.2)$$

Принять в качестве начального приближения корня $y_0 = X/2$. Итерационный процесс заканчивается, когда два последовательных приближения y_{n+1} и y_n будут удовлетворять условию

$$|y_{n+1} - y_n| < \varepsilon.$$

Если известно зачастую приближенное, даже очень грубое решение, то применение этого метода позволяет на каждом шаге итерационного процесса уточнить его и на каком-то этапе получить решение с требуемой точностью. Формула (6.2) порождает последовательность $y_1, y_2, y_3, \ldots, y_n, \ldots$

Можно показать, что если в качестве начального приближения y_0 будет выбрано любое положительное число, то итерационный процесс по формуле (6.2) будет сходиться к значению $\sqrt{X}$. Примем $y_0 = \frac{X}{2}$, как сказано в условии.

Сделаем одно общее замечание. В математике индексация переменных часто используется для того, чтобы различать время (номер шага) появления очередного значения (см. (6.2)). Так как в операторе присваивания

$$u := u + v$$

слева стоит "новое", $(i+1)$-е значение переменной u, а справа - "старое" (i)-е значение, то при алгоритмизации и программировании индексы в итерационной формуле не нужны: новые значения просто засылаются на старое место.

С учетом сделанных замечаний запишем алгоритм поставленной задачи извлечения квадратного корня. Прежде всего, учитывая, что для определения окончания вычислительного процесса на каждой итерации требуется проверка условия

$$|y_{n+1} - y_n| < \varepsilon \qquad (6.3)$$

преобразуем исходную формулу (6.2). Добавим и вычтем в правой части формулы y_n:

$$y_{n+1} = y_n + \frac{1}{2}\left(\frac{X}{y_n} + y_n\right) - y_n$$

или

$$y_{n+1} = y_n + \frac{1}{2}\left(\frac{X}{y_n} - y_n\right) \qquad (6.4)$$

Введем обозначения: $u = y_n$

$$v = \frac{1}{2}\left(\frac{X}{u} - u\right)$$

Тогда проверяемое условие (6.3) равносильно такому: $|v| < \varepsilon$.

Перейдем теперь непосредственно к алгоритму.

а) блок-схема

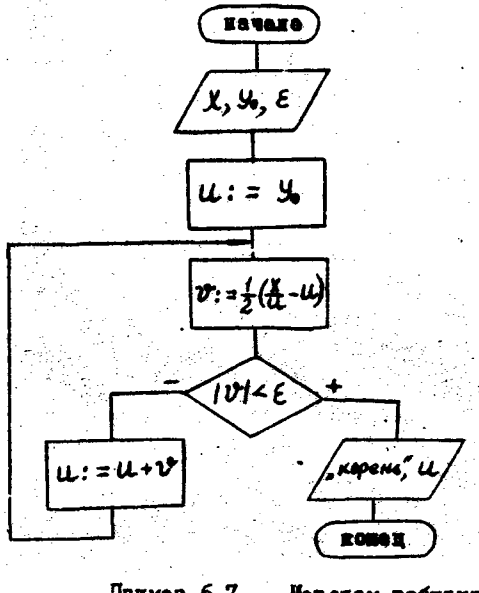


Рис. 6.9

б) построчная алгоритмическая запись

алгоритм НЬЮТОН

1. читать X, y_0, ε
2. $u := y_0$
3. $v := 0.5(X/u - u)$
4. если $|v| < \varepsilon$ идти к 6
5. $u := u + v$; идти к 3
6. записать "корень из", X, "=", u
7. конец

Пример 6.7. Методом табулирования исследовать предел функции:

$$\lim_{x \to 0} \frac{\sin x}{x}$$

Требуется:

а) установить предположительное значение предела;
б) исследовать поведение функции в окрестности точки
в) сделать вывод о скорости и характере сходимости.

Выполним табулирование на некотором промежутке $[A, B]$ с достаточно малым шагом H. B – величина, близкая к 0. В процессе исследования значения A, B, H изменяются, причем выбирать A, B и H надо так, чтобы обеспечить стремление X к нулю как справа, так и слева.

алгоритм ПРЕДЕЛ

1. читать A, B, H
2. $X := A$
3. записать "X", "$\sin X/X$"
4. $Y := \sin X/X$
5. записать X, Y
6. $X := X + H$
7. если $X \geq B$ идти к 4
8. конец

Результаты работы алгоритма при $A = 0.2$, $B = 0.01$, $H = -0.0$

X	$\sin X/X$
0.20	0.99335
0.19	0.99399
0.18	0.99461
0.17	0.99519
0.16	0.99574
0.15	0.99625
0.14	0.99674
0.13	0.99719
0.12	0.99760
0.11	0.99798
0.10	0.99833
0.09	0.99865
0.08	0.99893
0.07	0.99918
0.06	0.99940
0.05	0.99958
0.04	0.99973

X	$\sin X/X$
0.03	0.99985
0.02	0.99993
0.01	0.99998

Упражнения

Составить алгоритмы для решения следующих задач :

1) Составить таблицу значений функции $y = \frac{2\sin^2 x}{x^4+1}$ на отрезке $[a, b]$ с шагом h.

2) Вычислить сумму n первых членов последовательности с общим членом $a_k = \frac{k+1}{2k^2}$

3) Составить таблицы простых чисел табулированием трехчленов :

$$4x^2 - 162x + 1681$$
$$9x^2 - 249x + 1763$$

для $X = 1, 2, 3, \ldots, 40$.

4) Вычислить $y = a^k$, k – натуральное число.

5) Вычислить число π, используя бесконечное произведение:

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdots \frac{2n \cdot 2n}{(2n-1)(2n+1)}$$

6) Методом табулирования исследовать пределы функции :

$$\lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x ; \quad \lim_{x \to 0} \frac{1 - \cos x}{0{,}5x^2} ;$$

$$\lim_{x \to \infty} \left(\frac{x+1}{x-1}\right)^x ; \quad \lim_{x \to 0} \frac{x^5 - 1}{x^4 - 1} .$$

Установить предположительное значение предела, сделать вывод о скорости и характере сходимости.

7) Вычислить приближенное значение константы Эйлера :

$$C = \lim_{n \to \infty} \left\{ \sum_{k=1}^{n} \frac{1}{k} - \ln n \right\},$$

полагая значение n равным 1000, 2000, 10000.

8) Разработать алгоритм приближенного вычисления значения e^x по формуле

$$y = e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} \qquad (6.5)$$

Вычисления вести, пока очередное слагаемое не станет меньше заданной величины $\varepsilon > 0$.

Указание. Данная формула может быть представлена в виде циклического итерационного процесса, который описывается рекуррентными *) соотношениями между членами ряда y_i и частными суммами S_i:

$$y_{n+1} = y_n \cdot \frac{x}{n+1}$$

$$S_{n+1} = S_n + y_{n+1}$$

Из этих формул видно, что величины y_{n+1} и S_{n+1} могут быть получены на основе величин y_n, S_n, n, вычисленных в предыдущем "обороте" цикла.

Начальные значения этих величин $n = 0$, $S_0 = 1$, $y_0 = 1$ определяются на основе формулы (6.5). Так как разность между соседними значениями функции $S_{n+1} - S_n = y_{n+1}$, то момент окончания вычислений наступает, когда

$$|y_{n+1}| \le \varepsilon.$$

*) Рекуррентная формула (от латинского *recurrens* - возвращающийся) - зависимость, позволяющая выразить $(n+1)$-й член последовательности через значения ее первых n членов. В данном случае мы имеем дело с простейшими рекуррентными формулами, в которых $(n+1)$-й член выражается прямо через n-й.

Литература

1. Тихонов А.Н., Костомаров Д.П. Вводные лекции по прикладной математике - М.: Наука, 1984. - 190с.
2. Ершов А.П. ЭВМ в классе // Правда. - 1985. - 6 янв.
3. Ершов А.П. Монахов В.М. Основы информатики и вычислительной техники - М.: Просвещение, 1985. - 160с.
4. Основы алгоритмизации. Методические рекомендации к изучению школьного курса "Основы информатики и вычислительной техники" /сост. Лапчик М.П. - Омск, 1985. - 78с.

СОДЕРЖАНИЕ

Доп.Темплан, I986, поз.30

Владимир Сакович Макогон

Основы алгоритмизации в информатике

Отв.редактор Николай Яковлевич Тихоненко

редактор Бутвина Т.И.

БР 08308. Подп. к печати 8.08.86 г. Формат 60 x 84 1/16. Об'ем 4,5 п. л. Заказ № 3862. Тираж 500 экз. Цена 18 к. Гортипография Одесского облполиграфиздата, пер. № 3, Ленина, 49.