А. Г. КУШНИРЕНКО Г. В. ЛЕБЕДЕВ Р. А. СВОРЕНЬ

ОСНОВЫ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ПРОБНЫЙ УЧЕБНИК
ДЛЯ СРЕДНИХ УЧЕБНЫХ ЗАВЕДЕНИЙ**

Утверждено
Государственным комитетом СССР
по народному образованию

2-е издание

МОСКВА «ПРОСВЕЩЕНИЕ» 1991

ББК 73я72
К96

Кушниренко А. Г. и др.
К96 Основы информатики и вычислительной техники: Проб. учеб. для сред. учеб. заведений А. Г. Кушниренко, Г. В. Лебедев, Р. А. Сворень. – 2-е изд. – М.: Просвещение, 1991. – 224 с.: ил. – ISBN 5-09-003388-9

К $\frac{4306020000-119}{103(03)-91}$ **инф. письмо – 91, № 111** **ББК 73я72+32.973я72**

Учебное издание

Кушниренко Анатолий Георгиевич,
Лебедев Геннадий Викторович,
Сворень Рудольф Анатольевич

ОСНОВЫ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Пробный учебник
для средних учебных заведений

Зав. редакцией Т. А. Бурмистрова
Редактор А. К. Компанец
Художники А. Н. Горохов, В. В. Костин, Ю. В. Пахомов
Художественный редактор Ю. В. Пахомов
Технический редактор Л. М. Абрамова
Корректор Н. И. Новикова

ИБ № 13425

Подписано к печати с диапозитивов 07. 09. 90. Формат 60 × 90$^1/_{16}$. Бум. офсетная. Гарнит. литературная. Печать офсетная. Усл. печ. л. 14 + 0,25 форз. + 1 вкл. Усл. кр.-отт. 33,5. Уч.-изд.л. 12,06 + 0,42 форз. + 1,43 вкл. Цена 85 к.

Ордена Трудового Красного Знамени издательство «Просвещение» Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли. 129846, Москва, 3-й проезд Марьиной рощи, 41.

Отпечатано при посредстве В/О «Внешторгиздат»
Отпечатано Интердрук, Лейпциг, ГДР
Gedruckt bei Interdruck, Leipzig, DDR

ISBN 5-09-003388-9

ВВЕДЕНИЕ

Мы начинаем изучать новый предмет — информатику. ***Информатика*** изучает методы представления, накопления, передачи и обработки информации с помощью электронно-вычислительных машин (ЭВМ). Что же такое информация, что такое ЭВМ и как ЭВМ обрабатывает информацию?

§ 1. ИНФОРМАЦИЯ

1.1. ВЕЩЕСТВО, ЭНЕРГИЯ, ИНФОРМАЦИЯ — ВАЖНЕЙШИЕ СУЩНОСТИ НАШЕГО МИРА

Вещество — это все, что вокруг нас, это воздух и вода, горы и травы, хлеб и металл. Вещество — это то, что мы едим, чем дышим, из чего шьем одежды, делаем телевизоры и школьные тетради. Наконец, мы сами, наше тело, мускулы и нервы, кровь и кожа,— все это тоже вещество, атомы и молекулы.

Энергия приводит наш мир в движение. Энергия химических реакций дает силу мускулам, энергия солнечных лучей поднимает хлеба, электрическая энергия движет поезда и зажигает лампочки в наших домах.

Вещество и энергия — это две важнейшие сущности нашего мира, два важнейших его слагаемых. Но есть еще нечто столь же значительное и не менее важное для существования растений, животных, человека и человеческого общества в целом. Эта третья важнейшая сущность нашего мира — информация.

Информация — это не только сведения из книги, газетной заметки или передачи новостей, но и сведения, которые хранятся в рельефе ключа, в структуре сложной биологической молекулы, в радиосигналах, передаваемых на космический корабль. Информация в рельефе ключа позволяет открыть с его помощью определенный (''свой'') замок; информация в радиосигналах с Земли включает двигатель на космическом корабле и переводит корабль на другую орбиту; информация, запечатленная в структуре биологической молекулы, позволяет живой клетке производить определенные белки для новых тканей или для уничтожения попавших в организм микробов.

1. 2. ИНФОРМАЦИЯ И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

В каких процессах участвует информация, что с ней происходит в этих процессах? Каждый из нас слышал, что информацию можно собирать, хранить, передавать, обрабатывать и использовать. Глядя на дорогу, по которой мы идем, мы собираем информацию с помощью зрения. В нервной ткани глаза информация сложным образом преобразуется и передается в зрительные отделы головного мозга. Здесь она подвергается дальнейшей обработке и результат обработки немедленно используется: к нашим мышцам поступают сигналы (информация!) — и мы обходим лужи, перешагиваем через камни.

А вот еще одна система сбора и переработки информации в нашем организме — система терморегуляции. Наша кожа содержит около 300 тыс. клеток-датчиков, собирающих информацию о температуре тела. Собранная информация попадает в определенные участки головного мозга: центры теплоотдачи (управляют охлаждением тела) и теплопродукции (управляют нагреванием). В этих центрах информация обрабатывается и в случае необходимости нагрева, например, интенсивнее прокачивается теплая кровь по сосудам, а для охлаждения — усиливается процесс потоотделения. Если центры теплопродукции ''дезинформировать'' (например, раздражая их электрическим током), то они могут разогреть тело вплоть до смертельной температуры.

В каждом из огромного многообразия устройств и систем, созданных человечеством, в той или иной степени происходят сбор и обработка информации. Даже простейший автомат для продажи газированной воды, получив монету, собирает информацию о ней (вес, размеры), анализирует эту информацию и либо возвращает неподходящую монету, либо наливает стакан воды (с сиропом или без, в зависимости от монеты). Существуют уже и устройства, которые в толпе людей могут выделить нарушителей общепринятого порядка и задержать их! Именно так работают автоматы на входе в метро.

1.3. ИНФОРМАЦИЯ В ИСТОРИИ ОБЩЕСТВА

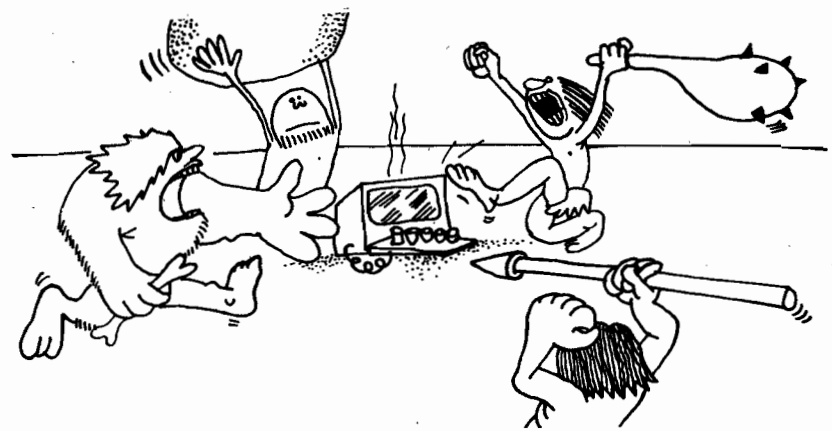
На протяжении всей своей истории человечество овладевало веществом, энергией и информацией. Целые эпохи в развитии человечества получили название по имени наиболее передовой технологии этой эпохи. Так, ''каменный век'' — это эпоха технологии обработки камня для получения орудий труда, ''бронзовый век''— эпоха овладения технологией обработки металла, ''век книгопечатания'' — эпоха овладения новыми методами распространения информации, ''век электричества'' — эпоха овладения новыми видами энергии. Еще двадцать — тридцать лет назад говорили, что наступил ''атомный век'', сейчас все чаще можно услышать о ''веке информации'' и ''веке ЭВМ''.

Потребность выразить и передать информацию привела к появлению речи, письменности, изобразительного искусства, вызвала к жизни книгопечатание, почтовую связь, телеграф, телефон, радио и телевидение. Почему же о ''веке информации'' заговорили только теперь, после появления ЭВМ?

Дело в том, что технология оказывает на общество революционное воздействие лишь тогда, когда она обретает всеобщий, универсальный характер. Так, подлинная революция в производстве, накоплении, передаче и обработке вещества началась с появления универсальных металлообрабатывающих станков и создания всеобщей транспортной системы (сети железных, автомобильных и других дорог). Аналогично, если в древних машинах (например, в ветряной или водяной мельнице) источник энергии был совмещен с самой машиной, то с развитием общества произошло разделение производства, передачи, потребления энергии. Революционные изменения в использовании энергии связаны прежде всего с появлением универсальных электрических машин и сетей передачи электроэнергии.

Появление ЭВМ — ***универсальной машины для обработки информации*** — означает начало революции в области накопления, передачи и обработки информации. Эта революция, следующая за революциями в овладении веществом и энергией, затрагивает и коренным образом преобразует не только сферу материального производства, но и интеллектуальные, ''духовные'' сферы жизни.

Рост производства ЭВМ, развитие информационных сетей, создание новых информационных технологий приводит к возникновению и росту информационных компонент во всех сферах жизни общества: в производстве, науке, образовании, медицине и т. д. Процесс перехода к информационному обществу охватил сегодня все развитые страны. Сердцевину, центральное ядро этого процесса составляет обработка информации на ЭВМ, которой посвящен наш учебник и к изучению которой мы приступаем.



Триада ''вещество, энергия, информация'' дает богатый материал для сравнений и аналогий. Отсутствие информационных сетей, например, можно сравнить с отсутствием электричества и бездорожьем. Компьютеризацию (***компьютер*** — это другое название ЭВМ) — с электрификацией и индустриализацией. Информатику — с физикой. Вопрос ''надо в школе учить программирование или учить пользоваться готовыми системами?'' можно пытаться переформулировать так: ''надо в школе изучать основы электричества (напряжение, ток, законы Ома) или учить включать/выключать свет и выворачивать лампочки?''. Конечно, такие аналогии не дают ответов на вопросы, но они позволяют лучше представить роль и место информатики в обществе.

На основе аналогий между веществом, энергией и информацией можно пытаться прогнозировать будущее, высказывать те или иные гипотезы. Раньше, например, количество выплавляемого страной металла (чугуна и стали) играло стратегическую роль, служило характеристикой развитости страны. Сейчас мы говорим об экономии металла, об экономии энергии, о неметаллоемких и неэнергоемких технологиях. Можно предположить, что то же будет происходить и с информацией — в будущем в обиход вполне могут войти слова ''малоинформационные технологии'' или ''борьба с информационным загрязнением окружающей среды''.

С другой стороны, роль информатики и ЭВМ в будущем обществе сейчас представить так же трудно, как было трудно представить роль электричества в конце XIX в. Вообразите, что произойдет, если сейчас всего на один день исчезнет электричество! А ведь изменения, которые несут ЭВМ, не менее глубоки и революционны.

1.4. ДВОИЧНОЕ КОДИРОВАНИЕ ИНФОРМАЦИИ. БИТ. БАЙТ

Одну и ту же информацию можно представить и передать по-разному. В старинном телеграфе, например, информация кодировалась и передавалась с помощью азбуки Морзе — в виде последовательностей из точек и тире. В книгах и газетах информация передается в виде текстов и изображений. Фразу, закодированную точками и тире азбуки Морзе, можно напечатать буквами на пишущей машинке, произнести, передать жестами и т. д.

В современной вычислительной технике информация чаще всего кодируется с помощью последовательностей сигналов всего двух видов: намагничено или не намагничено, включено или выключено, высокое или низкое напряжение и т. д. Принято обозначать одно состояние цифрой 0, а другое — цифрой 1. Такое кодирование называется ***двоичным кодированием,*** а цифры 0 и 1 называются ***битами*** (от англ. bit — binary digit — двоичная цифра).

При двоичном кодировании текстовой информации каждому символу сопоставляется его ***код*** — последовательность из

фиксированного количества нулей и единиц. В большинстве современных ЭВМ каждому символу соответствует последовательность из 8 нулей и единиц, называемая ***байтом*** (англ. byte). Всего существует 256 разных последовательностей из 8 нулей и единиц — это позволяет закодировать 256 разных символов, например большие и малые буквы русского и латинского алфавитов, цифры, знаки препинания и т. д. Соответствие байтов и символов задается с помощью таблицы, в которой для каждого кода указывается соответствующий символ. Вот фрагмент такой таблицы для кодировки КОИ-8, широко распространенной в СССР:

Код	Символ	Код	Символ	Код	Символ	Код	Символ
00100000	пробел	00110000	0	01000000	@	01010000	P
00100001	!	00110001	1	01000001	A	01010001	Q
00100010	"	00110010	2	01000010	B	01010010	R
00100011	#	00110011	3	01000011	C	01010011	S
00100100	$	00110100	4	01000100	D	01010100	T
00100101	%	00110101	5	01000101	E	01010101	U
00100110	&	00110110	6	01000110	F	01010110	V
00100111	'	00110111	7	01000111	G	01010111	W
00101000	(	00111000	8	01001000	H	01011000	X
00101001	)	00111001	9	01001001	I	01011001	Y
00101010	*	00111010	:	01001010	J	01011010	Z
00101011	+	00111011	;	01001011	K	01011011	[
00101100	,	00111100	<	01001100	L	01011100	\
00101101	—	00111101	=	01001101	M	01011101	]
00101110	.	00111110	>	01001110	N	01011110	^
00101111	/	00111111	?	01001111	O	...	

Коду 00100000 в этой таблице соответствует ***пробел*** — пустой промежуток величиной в один символ, который используется для отделения одного слова от другого.

Коды русских букв отличаются от кодов латинских. Например, большая русская буква ''М'' имеет код 11101101, буква ''И'' — код 11101001, буква ''Р'' — код 11110010, буква ''У'' — код 11110101. Таким образом, слово ''МИР'' кодируется последовательностью из 24 бит

111011011110100111110010,

а фраза ''МИРУ МИР'' — последовательностью

111011011110100111110010111110101
001000001110110111101001111110010.

Последовательностями нулей и единиц можно закодировать и графическую информацию. Всмотритесь в газетную фотографию и вы увидите, что она состоит из мельчайших точек. У разного

полиграфического оборудования густота этих точек разная — фотографии в "Правде" намного четче, чем в "За рубежом". В большинстве газет фотографии содержат 24 точки на сантиметр длины, т. е. фотография размером 10×10 сантиметров состоит примерно из 60 тыс. точек. Если это только черные и белые точки, то каждую из них можно закодировать 1 битом, а всю фотографию — последовательностью из 60 тыс. бит. Если точки бывают разные, то одним битом для точки не обойтись. Два бита позволяют закодировать 4 оттенка точек: 00 — белый цвет, 01 — светло-серый, 10 — темно-серый, 11 — черный. Три бита позволяют закодировать 8 оттенков и т. д.

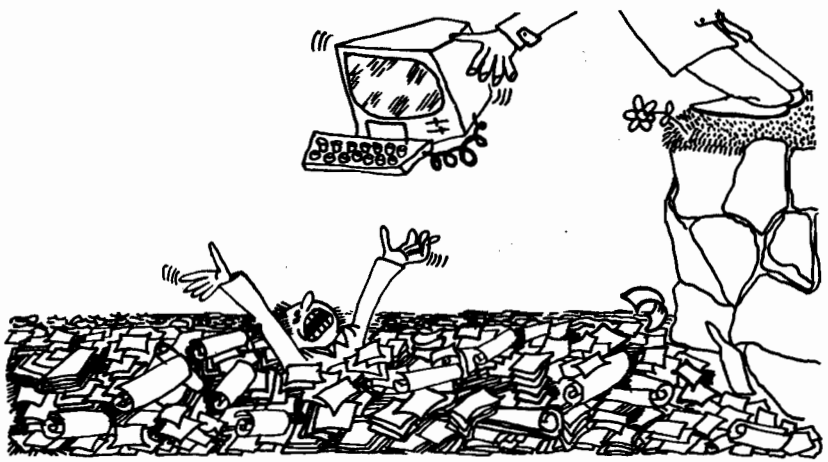
1.5. ЕДИНИЦЫ ИЗМЕРЕНИЯ ИНФОРМАЦИИ

Для измерения длины, массы, времени, силы тока и т. д. придуманы приборы и процедуры измерения. Чтобы узнать длину стержня, достаточно приложить к нему линейку с делениями, силу тока можно измерить амперметром.

А как узнать количество информации в сообщении, в каких единицах эту информацию измерять? Для двоичных сообщений в качестве такой числовой меры используется количество бит в сообщении. Это количество называют ***информационным объемом*** сообщения. Например, сообщение "МИРУ МИР" в коде КОИ-8 имеет информационный объем 8 байт (64 бит).

Биты и байты используются также для измерения "емкости" памяти и для измерения скорости передачи двоичных сообщений. Скорость передачи измеряется количеством передаваемых бит в секунду (например, 19200 бит/с).

Наряду с битами и байтами для измерения количества информации в двоичных сообщениях используются и более крупные единицы:

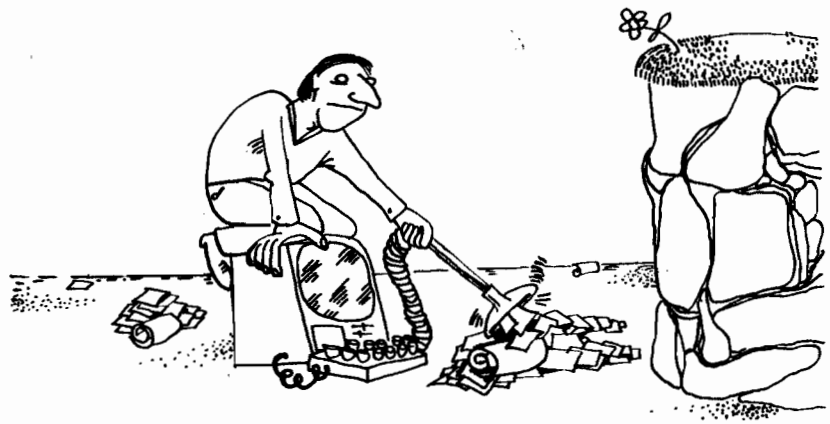


1 Кбит (один килобит) $= 2^{10} = 1024$ бит ($\approx$ 1 тыс. бит);
1 Мбит (один мегабит) $= 2^{20} = 1048576$ бит ($\approx$ 1 млн. бит);
1 Гбит (один гигабит) $= 2^{30} \approx 10^9$ бит (миллиард бит);
1 Кбайт (один килобайт) $= 2^{10} = 1024$ байт ($\approx$ 1 тыс. байт);
1 Мбайт (один мегабайт) $= 2^{20} = 1048576$ байт ($\approx$ 1 млн. байт);
1 Гбайт (один гигабайт) $= 2^{30}$ ($\approx$ 1 млрд. байт).

К единицам измерения многих физических величин мы привыкли, и нам не нужно пояснять, что такое 1 миллиметр или 10 километров. А бит, байт, килобайт, мегабайт, гигабайт — много это или мало?

В коде КОИ-8 каждая буква, знак препинания, пробел — это 1 байт. На страницу нашего учебника помещается чуть меньше 50 строк, в каждой строке — примерно 60 знаков (60 байт). Таким образом, полностью заполненная текстом страница учебника имеет информационный объем около 3000 байт (3 Кбайт). Средняя страница содержит около 2.5 Кбайт, а весь учебник — чуть больше 0.5 Мбайт текста. В Большой Советской Энциклопедии примерно 120 Мбайт. В одном номере четырехстраничной газеты — 150 Кбайт, а если собрать по одному номеру всех газет, выходящих в нашей стране, то в них будет уже гигабайт информации. Если человек говорит по 8 часов в день без перерыва, то за 70 лет жизни он наговорит около 10 гигабайт информации (это 5 млн. страниц — стопка бумаги высотой 500 м).

Один черно-белый телевизионный кадр (при 32 градациях яркости каждой точки) содержит примерно 300 килобайт информации. Цветной кадр, образованный из трех кадров основных цветов (красный, синий, зеленый), содержит уже около мегабайта информации. А 1,5-часовой цветной телевизионный художественный фильм (при частоте 25 кадров в секунду) — 135 гигабайт.



Если на условной шкале изобразить бит примерно одним миллиметром (точнее, 1.25 мм), то байт в этом масштабе будет представлен сантиметром, килобайт — десятиметровым отрезком, мегабайт — десятикилометровым отрезком, ну а гигабайт вытянется в 10 000 километров — это расстояние от Москвы до Владивостока. Как видите, диапазон, который охватывают единицы измерения информации, очень велик.

1.6. ИНФОРМАЦИЯ — ПЕРВИЧНОЕ, НЕОПРЕДЕЛЯЕМОЕ ПОНЯТИЕ ИНФОРМАТИКИ

Обычно слово ''информация'' ассоциируется у нас со смыслом, значимостью сообщения. С этой точки зрения телеграмма о дате приезда не несет никакой информации, если мы эту дату уже знаем. Смысл и значимость, однако, понятия человеческие, субъективные. Информатика же изучает обработку информации с помощью ЭВМ, т. е. с научно-технической точки зрения. Поэтому понятие информации в информатике отличается от его обыденной трактовки.

Введенный выше информационный объем двоичного сообщения, например, является чисто технической характеристикой и никак не связан с его смыслом и значимостью (подобно тому, как вес груза не связан с его ценностью). Бессмысленное сообщение ''вораплвыорпалвывцупгкшнвеывапорчбьютизкгмяхзщншапждр любсьт'' имеет информационный объем 58 байт — больше, чем сообщение ''готово'' (6 байт).

Так что же такое ''информация''? Увы! — этот термин в информатике является первичным, неопределяемым. Отсутствие строгого определения, однако, не мешает нам измерять объем информации и обрабатывать ее, подобно тому, как отсутствие строгого определения прямой и точки в планиметрии не мешает нам рисовать треугольники, доказывать теоремы и решать задачи. Информации и ее обработке на ЭВМ посвящен весь наш курс, а пока можно представлять себе порцию информации как последовательность битов (0 и 1) или байтов (символов).

1.7. ОБРАБОТКА ИНФОРМАЦИИ

Под ***обработкой информации*** в информатике понимают ***любое преобразование информации*** из одного вида в другой, ***производимое по строгим формальным правилам.*** Примерами таких преобразований могут служить: замена одной буквы на другую в тексте; замена нулей на единицы, а единиц на нули в последовательности битов; сложение двух чисел, когда из информации, представляющей слагаемые, получается результат — сумма.

Слова ''обработка информации'', таким образом, вовсе не подразумевают восприятие информации или ее осмысление.

ЭВМ — всего лишь машина и способна только к технической, машинной обработке информации.

Конечно, технические преобразования информации обычно производятся с целью достижения некоторого осмысленного эффекта. Например, если в тексте восклицательный знак заменить на вопросительный, то это будет соответствовать и некоторому смысловому изменению. Однако сама замена восклицательного знака на вопросительный носит технический характер и может быть произведена в любом тексте:

Это правда! ⟶ Это правда?
a+%539—!(рол ⟶ a+%539—?(рол

Обработка информации на ЭВМ обычно состоит в выполнении огромного количества такого рода элементарных, технических операций.

УПРАЖНЕНИЯ

1. Приведите несколько бытовых примеров получения, хранения, передачи, обработки, использования информации.
2. Сколько существует различных последовательностей из 4 нулей и единиц?
3. Закодируйте в коде КОИ-8: а) слово ''РИМ''; б) текст ''I LOVE YOU''; в) текст ''TO BE OR NOT TO BE''; г) число ''1990''; д) выражение (последовательность символов) ''2X+Y=0''.
4. Какие последовательности символов закодированы следующими кодами в КОИ-8.
   а) 111011011110100111101101;
   б) 001100010011011000110001;
   в) 010000010101100000101011010000100011110100110000;
   г) 010000110100111101001101010100000101010101010100010001010101010010?
5. Считая, что каждый символ кодируется одним байтом, определите приблизительно информационный объем:
   а) вашего школьного дневника;
   б) годовой подшивки газеты ''Комсомольская правда'';
   в) учебников в вашем портфеле;
   г) классного журнала.
6. Сколько двоичных цифр (бит) необходимо, чтобы закодировать одну школьную оценку?
7. Человек способен различить примерно 100 градаций яркости. Сколько бит необходимо, чтобы их закодировать?
8. Придумайте эксперимент, позволяющий узнать, сколько символов в секунду вы читаете, пишете, произносите. Проведите этот эксперимент.
9. Подсчитайте приблизительно, сколько байт вы прочли, написали, произнесли за время учебы в школе.
10. Придумайте способ передать изображение по телефону. Сколько времени понадобится, чтобы передать фото 4×6 см?

11. В приведенных ниже последовательностях каждый следующий элемент получен по некоторому строгому правилу. Угадайте это правило:

а) а, б, в, г, д, е, ... ;
б) 1, 2, 3, 4, 5, 6, 7, ... ,
в) 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, ... ;
г) 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... ;
д) 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 0, 1, 1, 1, 2, 1, ... ;
е) победа, обеда, беда, еда, ... ;
ж) о, д, т, ч, п, ш, с, в, д, д, ... ;
з*) 1, 11, 21, 1211, 111221, 312211, 13112221, ...

12. Петя и Коля играют в следующую игру: Петя задумывает правило преобразования текстовой информации. Коля может задавать Пете любые тексты и узнавать результаты преобразования. Задача Коли — отгадать это правило. Ниже приведены вопросы Коли и ответы Пети в нескольких таких играх. Попробуйте отгадать, какое правило задумал Петя в каждой игре:

а) А → Б; мама → нбнб; ЭВМ → ЮГН; язык → аиьл;
б) А → 1; мама → 4; ЭВМ → 3; язык → 4;
в) А → 0+1; мама → 2+2; ЭВМ → 2+1; язык → 2+2;
г) А → А; мама → амам; ЭВМ → ВМЭ; язык → зыкя.

13. В условиях задачи 12 Петя задумывает правило преобразования целых чисел. Ниже приведены вопросы Коли и ответы Пети в нескольких таких играх. Попробуйте отгадать, какое правило задумал Петя в каждой игре:

а) 1 → 0; 5 → 4; 0 → −1; 1990 → 1989;
б) 1 → 0; 2 → 0; 10 → 9; 3 → 3; 20 → 18; 1990 → 1989;
в) 1 → 1; 7 → 1; 10 → 2; 187 → 3; 1990 → 4;
г*) 1 → 0; 8 → 2; 16 → 1; 1990 → 3; 1989 → 4; 100 → 2; 108 → 3; 6 → 1; 7 → 0; 23 → 0; 50 → 1.

14. Опишите правило преобразования текста, которое фразы вида ''Тебя зовут Вася?'' превращает во фразы вида ''Это тебя зовут Вася!''. Примените это преобразование к нескольким другим фразам.

15. Преобразование информации меняет в тексте на английском языке каждое сочетание ''ing'' на ''ed''. Приведите примеры текстов, для которых такое преобразование приводит к а) осмысленному, б) бессмысленному результату.

16. Даны: лист миллиметровки, иголка, лупа с десятикратным увеличением. Придумайте способ записать на листе миллиметровки как можно больше информации.

17. Даны: лист тонкого картона и тупое шило. Придумайте способ записать на листе информацию так, чтобы ее можно было считать в темноте, ''на ощупь''.

18. Коля сидит дома, делает уроки и не подходит к телефону. Петя набирает Колин номер, ждет некоторое время и кладет трубку. Затем снова набирает Колин номер и т. д. Придумайте способ передачи информации с помощью таких звонков.

§ 2. ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ

2.1. КРАТКАЯ ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

С древнейших времен человек конструирует себе в помощь различные приспособления для облегчения вычислений. Еще в V в. до н. э. греки и египтяне использовали абак — устройство, похожее на русские счеты.

В 40-х годах XVII в. один из крупнейших ученых в истории человечества — математик, физик, философ и богослов Блез Паскаль изобрел и изготовил механическое устройство, позволяющее складывать числа.

Механическое устройство, позволяющее не только складывать числа, но и умножать их, было изобретено другим великим математиком и философом — Готфридом Вильгельмом Лейбницем в конце XVII в. В работах Лейбница шла речь и о механическом устройстве, которое может оперировать со словами и понятиями.

Наряду с устройствами, предназначенными для вычислений, развивались и механизмы для ***автоматической работы по заданной программе*** (музыкальные автоматы, шарманки, часы с боем и т. п.). В шарманку, например, помещали диски с по-разному расположенными штырьками — в зависимости от расположения штырьков звучала та или иная мелодия. В ткацком станке Жаккарда узор ткани задавался с помощью дырочек в тонких картонных картах (перфокартах). Для смены узора достаточно было по-другому пробить дырочки в перфокарте.

Важный прогресс в области вычислительной техники связан с именем Чарльза Беббиджа (середина XIX в.). Соединив идею механической арифметической машины Лейбница с идеей программного управления, Беббидж разработал проект машины, названной им ''аналитической''. Этот проект не был реализован, однако по своим возможностям машина Беббиджа не уступала первым ЭВМ: в ней была предусмотрена память для хранения 1000 чисел по 50 десятичных знаков; арифметические операции выполнялись в соответствии с программой, записанной на жаккардовых перфокартах. В программе можно было задать автоматическое повторение группы арифметических операций, а также выполнение группы операций только при определенном условии. С машиной Беббиджа связано появление профессии программиста. Первым программистом мира стала дочь поэта Дж. Байрона Ада Лавлейс. Программы, написанные Адой Лавлейс, предназначались для вычисления значений некоторых числовых функций.

В конце 30-х годов XX в. американцы Дж. Атанасов (болгарин по происхождению) и К. Берри построили ЭВМ, включавшую в себя электронную память и электронное устройство сложения и вычитания, а также ряд механических компонент. Эта машина еще не была универсальной, однако область ее применения значи-

тельно превосходила область применения механических арифмометров. В 1942 г. была построена усовершенствованная модель ЭВМ Атанасова — Берри, предназначенная прежде всего для решения систем линейных уравнений (до 30 уравнений с 30 неизвестными).

До конца 40-х годов был создан ряд других машин, более мощных и совершенных. В 1946 г. в научной статье трех американских авторов — Дж. фон Неймана, Г. Голдстайна, А. Бернса — были изложены основные принципы построения универсальной ЭВМ, использующей одну и ту же память и для хранения обрабатываемых данных, и для хранения программы вычислений. Первая машина, реализующая эти принципы,— ЭВМ EDSAC — была построена в 1949 г. под руководством М. В. Уилкса в Англии, в Кембриджском университете (в нем когда-то учился Беббидж). Через год была построена универсальная ЭВМ EDVAC в США.

Основоположником отечественной вычислительной техники стал Сергей Алексеевич Лебедев. Под его руководством были созданы первые отечественные ЭВМ: в 1951 г. в Киеве — МЭСМ (Малая Электронная Счетная Машина) и в 1952 г. в Москве — БЭСМ (Быстродействующая Электронная Счетная Машина). Почти одновременно с БЭСМ были разработаны ЭВМ М-2 и "Стрела".

С середины 50-х годов начался бурный рост вычислительной техники. Сейчас в мире уже работает свыше 50 млн. персональных компьютеров, десятки миллионов встроенных, игровых, домашних. Степень развития общества начинает определять не энерговооруженность, а "инфовооруженность": количество и производительность ЭВМ в расчете на одного работающего, наличие выхода на мировые сети коммуникаций и т. д.

2.2. ОСНОВНЫЕ КОМПОНЕНТЫ ЭВМ

В состав любой ЭВМ входят ***процессор, память, устройства ввода и вывода информации.***

Процессор занимается непосредственно обработкой информации — это своего рода "мозг" ЭВМ. Основными характеристиками процессора являются ***быстродействие*** (число выполняемых операций в секунду) и ***разрядность.*** Разрядность характеризует объем информации, который процессор обрабатывает за одну операцию: 8-разрядный процессор за одну операцию обрабатывает 8 бит (1 байт) информации, 32-разрядный — 32 бита.

В ***оперативной памяти (ОЗУ)*** ЭВМ в двоичном виде запоминаются обрабатываемая информация, программа ее обработки, промежуточные данные и результаты работы. Кроме оперативной памяти, у ЭВМ может быть ***постоянная память (ПЗУ),*** содержимое которой устанавливается на заводе-изготовителе и в дальнейшем не изменяется. Основной характеристикой памяти является ее объем (количество запоминаемой информации).

Устройства ввода и вывода обеспечивают ввод информации в память ЭВМ и выдачу ее наружу, т. е. обмен информацией с

внешним миром. Для школьных ЭВМ важнейшее из этих устройств — ***монитор,*** или ***экран.*** Изображение на экране строится из отдельных точек. Чем больше этих точек, тем выше качество изображения на экране. Для хранения изображения используется специальная память — ***видеопамять.***

Вот цифры, характеризующие быстродействие, разрядность, объем ОЗУ и видеопамяти школьных и некоторых других ЭВМ:

Название ЭВМ	Быстродействие (тыс. оп/сек)	Разрядность (бит)	Объем ОЗУ (Кбайт)	Видеопамять (Кбайт)
Корвет	600	8	64	48
УК НЦ*	800	16	64	96
Ямаха MSX—1	1000	8	64	16
Ямаха MSX—2	1000	8	128	128
БК—0010	600	16	16	16
IBM PC/XT	1000	16	640	16—256

* В ЭВМ УК НЦ есть еще один, так называемый "периферийный" 16-разрядный процессор быстродействием 600 тыс. оп/с и объемом ОЗУ 32 Кбайта.

2.3. ВСТРОЕННЫЕ ЭВМ

ЭВМ может быть встроена в различное промышленное, научное, военное, бытовое оборудование и осуществлять автоматическое управление этим оборудованием. Например, ЭВМ, встроенная в систему зажигания легкового автомобиля, постоянно подбирает наиболее выгодный момент зажигания горючей смеси в цилиндрах двигателя в зависимости от числа оборотов двигателя, температуры охлаждающей жидкости, положения педали "газа" и ряда других параметров. Устройствами ввода информации для этой ЭВМ являются датчик числа оборотов двигателя, датчик температуры и другие, а выходная информация поступает в блок, управляющий зажиганием. В памяти ЭВМ находятся программа работы процессора по выбору оптимального момента зажигания, текущая информация, полученная от датчиков, а также промежуточные результаты работы процессора по программе.

2.4. ПЕРСОНАЛЬНЫЕ ЭВМ

Персональные ЭВМ предназначены для работы в диалоге с человеком. Устройства ввода и вывода такой ЭВМ обеспечивают удобный обмен информацией между человеком и ЭВМ — это ***клавиатура***; ***"мышь"***; ***монитор*** (или ***экран***); ***печатающее устройство*** (или ***принтер***); ***внешняя память.***

Клавиатура предназначена для передачи информации от человека к ЭВМ и похожа на клавиатуру обычной пишущей машинки.

"Мышь" (фото 28 вклейки) также предназначена для передачи информации от человека к ЭВМ. ''Мышь'' — это небольшая коробочка, которую человек может перемещать по плоскости стола и которая при этом перемещении посылает сигналы в ЭВМ. С помощью этого устройства можно, например, вводить в ЭВМ картинки — для этого достаточно положить картинку на стол и обвести ее ''мышкой''.

Монитор предназначен для передачи человеку информации (текстов и изображений) от ЭВМ. Грубо говоря, монитор — это обычный телевизор, изображение на котором строит ЭВМ.

Печатающее устройство предназначено для передачи информации от ЭВМ к человеку в виде текстов и изображений, напечатанных на бумаге.

Внешняя память на магнитных дисках предназначена для запоминания и воспроизведения больших объемов информации. Внешняя память позволяет также хранить информацию в те периоды, когда ЭВМ выключена. Внешняя память на магнитных дисках устроена на тех же принципах, что и обычный магнитофон, только вместо кассет с пленкой в качестве носителя информации используются ***магнитные диски*** — круглые пластинки, покрытые слоем магнитного вещества (фото 26 вклейки). Внешняя память на магнитных дисках позволяет хранить информацию отдельно от ЭВМ и переносить информацию с одной ЭВМ на другую.

Если персональную ЭВМ через ***модем*** подключить к обычной телефонной сети, то она сможет обмениваться информацией с другими ЭВМ. При передаче информации по телефону модем превращает последовательности электрических импульсов в последовательности звуковых сигналов разного тона. При приеме модем проделывает обратное преобразование.

Кроме того, ЭВМ может содержать устройства ввода и вывода для получения информации от разного рода датчиков и выработки сигналов управления различными манипуляторами, роботами и другими исполнительными устройствами (или коротко ***исполнителями***). Примером такого устройства может служить графопостроитель, который по командам ЭВМ строит изображение тушью на листе бумаги или краской на листе пластмассы.

2.5. ПОТОКИ ИНФОРМАЦИИ ПРИ РАБОТЕ ШКОЛЬНОЙ ЭВМ

В СССР серийно выпускаются не отдельные школьные ЭВМ, а комплекты учебной вычислительной техники (КУВТ). В комплект входит одна ЭВМ преподавателя и несколько соединенных с нею ЭВМ учеников.

Составные части ЭВМ преподавателя и их взаимодействие в процессе работы изображены на рисунке 1.

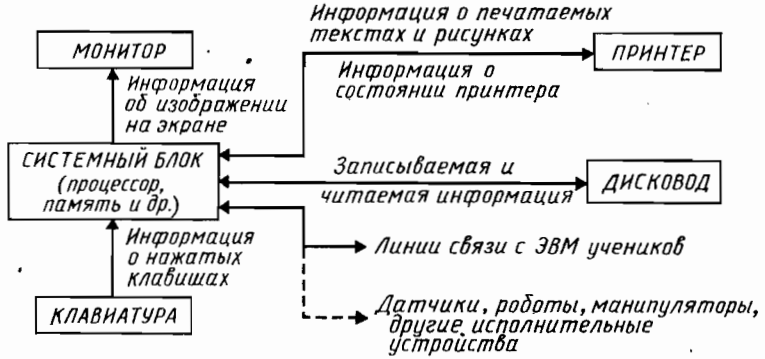


Рис. 1

Клавиатура передает информацию о нажатых клавишах в системный блок.

На экране ***монитора*** создается изображение, информация о котором поступает из системного блока.

Принтер (печатающее устройство; от англ. print — печатать) печатает на бумаге информацию, поступающую к нему из системного блока. В системный блок от принтера поступает информация о его состоянии: включено ли питание, вставлена ли бумага и т. д.

Дисковод (внешнее запоминающее устройство, или ***внешняя память***) используется для записи информации на гибкий магнитный диск — ***дискету,*** а также для считывания записанной туда информации.

Основные устройства ЭВМ — процессор и память — входят в состав ***системного блока.*** Он хранит и обрабатывает информацию. В школьной ЭВМ он размещается в одном корпусе с клавиатурой, под клавишами. В системном блоке есть свободные разъемы, позволяющие подключить другие устройства, например, ''мышь'' или модем.

УПРАЖНЕНИЯ

1. Приведите несколько примеров использования ЭВМ. С какими устройствами связана ЭВМ в этих примерах?
2. Приведите несколько примеров задач, для решения которых, на ваш взгляд, стоило бы применить ЭВМ. Какие исполнительные устройства для этого нужны? Что даст применение ЭВМ в этих задачах?
3. Оперативная память ЭВМ ''Корвет'' имеет объем 64 Килобайта (т. е. может запомнить 64 Кбайт информации). Сколько примерно страниц нашего учебника можно уместить в эту память?

4. Изображение на экране монитора ЭВМ ''Корвет'' состоит из 256 строк по 512 точек каждая. Какой объем памяти необходим для запоминания одного черно-белого изображения без полутонов, т. е. когда каждая точка может быть либо белой, либо черной?

5. Каждая точка на экране монитора ЭВМ ''Корвет'' может быть окрашена в 8 различных цветов (или иметь одну из 8 градаций яркости, если изображение нецветное). Какой объем памяти необходим для запоминания одного цветного изображения? Для запоминания трех разных изображений? Хватит ли объема ОЗУ ЭВМ ''Корвет'' для запоминания 10 цветных изображений?

6. Сменный гибкий магнитный диск (дискета) внешней памяти на гибких магнитных дисках ЭВМ ''Корвет'' может хранить 720 Кбайт информации. Определите приблизительно:

а) сколько страниц нашего учебника можно запомнить на одной дискете;

б) сколько дискет нужно для хранения всего учебника целиком;

в) сколько разных цветных изображений можно уместить на одну дискету;

г) сколько номеров газеты ''Правда'' (без учета иллюстраций) можно уместить на одну дискету.

7. Учебный класс, содержащий 12 ученических и 1 учительскую ЭВМ ''Корвет'', стоит 25 000 р. и рассчитан на эксплуатацию по 42 ч в неделю в течение 10 лет. Стоимость технического обслуживания, ремонта и эксплуатации класса составляет 1200 р. в год. Класс в целом потребляет мощность 1.5 кВт, цена 1 кВт·ч электроэнергии равна 4 к. Определите стоимость работы на одной ученической ЭВМ в течение одного урока в этом классе, считая, что класс исправно работает все 10 лет.

8. Печатающее устройство (принтер) ЭВМ ''Корвет'' печатает 80 символов в секунду. Сколько примерно времени нужно, чтобы напечатать целиком весь наш учебник?

9. Информация от машины учителя к машине ученика в классе ЭВМ ''Корвет'' передается со скоростью 9600 бит/с. Сколько времени нужно, чтобы переслать содержимое памяти ЭВМ учителя ученику? Сколько времени нужно, чтобы переслать одно цветное изображение?

10. В течение урока 12 учеников пишут диктант — каждый на своей ученической ЭВМ ''Корвет''. Оцените приблизительно, сколько минут понадобится, чтобы переслать поочередно работы всех учеников на ЭВМ учителя. Уместятся ли все эти работы на одной дискете?

11. Процессор ЭВМ ''Корвет'' делает около 600 тыс. оп/с. Для чтения из ОЗУ и проверки совпадения двух символов нужно около 10 операций. Сколько времени понадобится, чтобы убедиться в полном побуквенном совпадении двух диктантов в ОЗУ?

§ 3. ОБРАБОТКА ИНФОРМАЦИИ НА ЭВМ

3.1. ПРОГРАММИРОВАНИЕ КАК ВИД ЧЕЛОВЕЧЕСКОЙ ДЕЯТЕЛЬНОСТИ

Рассмотрим управление каким-нибудь исполнительным устройством (***исполнителем***), например станком, роботом, самолетом, магнитофоном, кухонным комбайном или ядерным реактором. Управлять можно по-разному. Можно, например, выполнить какое-то действие, посмотреть на результат, решить, что делать дальше, снова выполнить какое-то действие и т. д. Именно так операторы управляют ядерными реакторами, капитаны — пароходами, летчики — самолетами.

Часто, однако, такой способ действий нежелателен или просто невозможен: работа может происходить в опасной среде, вдали от человека; скорость реакции человека может оказаться недостаточной; работа может быть монотонной и приводить к катастрофическим ошибкам человека и т. п. В таких случаях вместо анализа ситуации и принятия решений ''по ходу'', в процессе управления можно заранее составить полный план (***программу***) будущего управления (т. е. записать, при каких условиях, какие действия и в какой последовательности должно будет выполнить устройство), а затем заставить устройство выполнить эту последовательность действий автоматически. Такой вид деятельности называется ***программированием*** (или ***алгоритмизацией***).

Программирование как вид человеческой деятельности требует специфического ***алгоритмического стиля мышления.*** В отличие от управления тут не так важна скорость реакции или наблюдательность человека. Основу алгоритмического стиля мышления составляют умение планировать длинные последовательности действий, ясно и четко записывать их в виде программ, умение предвидеть их последствия и предусмотреть все условия, которые могут возникнуть при выполнении программы.

3.2. ОБРАБОТКА ИНФОРМАЦИИ НА ЭВМ

Любая ЭВМ умеет выполнять небольшое количество очень простых команд, например: заменить нолик на единичку, записать в байт восемь ноликов, сложить или вычесть пару чисел. Секрет могущества ЭВМ в том, что она может быстро выполнять длинные последовательности таких команд и обрабатывать большие объемы информации.

Описание того, какие действия, в каком порядке и над какими порциями информации должна произвести ЭВМ, называется ***программой для ЭВМ.*** Образно говоря, ЭВМ без программы представляет собой груду железа, не способную ни к какой переработке информации,— она бесполезна так же, как магнитофон

без кассет. Поэтому составление программ — программирование — составляет основу любого использования ЭВМ.

Сфера применения конкретной ЭВМ определяется не столько ее конструкцией, сколько набором созданных для нее программ. Одна и та же ЭВМ может быть использована в системе управления ракетой и в медицинском приборе; с ЭВМ, целыми днями рассчитывающей зарплату, можно вечером поиграть в шахматы и т. п. Для того чтобы переключить ЭВМ с одной работы на другую, достаточно сменить программу в ее памяти.

3.3. ЯЗЫК ПРОГРАММИРОВАНИЯ

Для того чтобы ЭВМ могла выполнить программу, программа должна быть записана по строгим правилам, в виде, доступном для обработки на ЭВМ. Такой набор правил называется ***языком программирования*** или ***алгоритмическим языком.*** Воспринимая программу, записанную на языке программирования, ЭВМ некоторым образом ее преобразует, помещает в собственную память, а затем выполняет команды, соответствующие данной программе.

ЭВМ может понимать несколько разных языков программирования. Вот, например, как выглядит программа, которая вычисляет и выводит на экран сумму квадратов чисел от 1 до 100, на школьном алгоритмическом языке и языках программирования Бейсик и Паскаль:

```
алг сумма квадратов
нач цел s, i
    s:=0
    нц для i от 1 до 100
    |   s:=s+i*i
    кц
    вывод "Сумма квадратов=", s
кон
```

а) школьный алгоритмический язык

```
10 S=0
20 FOR I=1 TO 100
30  S=S+I*I
40 NEXT I
50 PRINT  "Сумма  квадратов=";S
60 STOP
```

б) Бейсик

```
program sum;
 var s,i : integer;
begin
  s:=0;
  for i:=1 to 100 do s:=s+i*i;
  writeln ('Сумма квадратов=', s)
end.
```

в) Паскаль

Вы можете видеть, что программа на языке программирования записывается в виде текста, содержащего буквы, цифры, знаки операций и т. д. Любой такой текст, как мы знаем, можно закодировать, например, с помощью последовательности байтов. Таким образом программу, как и любую другую информацию, можно закодировать, передать, поместить в память ЭВМ и т. п.

Для ввода программы в память ЭВМ ее можно набрать на клавиатуре, считать с магнитного диска или передать по линиям связи с другой ЭВМ. В памяти ЭВМ хранится как сама программа, так и обрабатываемые по этой программе данные. При выполнении ЭВМ анализирует программу и в соответствии с программой выполняет те или иные преобразования над данными. По завершении выполнения программы результаты ее выполнения могут быть изображены на экране, напечатаны на бумаге, записаны на дискету или другим образом выведены из ЭВМ с помощью одного из устройств вывода.

3.4. ОТДЕЛЕНИЕ ИНФОРМАЦИОННОГО ПРОИЗВОДСТВА ОТ МАТЕРИАЛЬНОГО

Программирование как вид человеческой деятельности (т. е. как составление плана будущих действий) появилось задолго до появления ЭВМ. Кулинарные и химические рецепты, схемы поиска кладов, должностные инструкции — простейшие примеры «программ». Элементы программирования использовались и при создании станков-автоматов или полуавтоматов, которые изготавливают детали, автоматически выполняя заданную последовательность действий. Однако до появления ЭВМ переход от изготовления одной детали к другой (т. е. смена программы обработки детали) был длительным и трудоемким — требовалось проектирование и изготовление новых станков.

Появление ЭВМ кардинально изменило эту ситуацию. Станок-автомат оказалось возможным заменить на пару: 1) универсальный станок, выполняющий элементарные технологические операции, и 2) ЭВМ, которая по заданной программе заставляет станок выполнять те или иные этапы обработки детали. Массовое изготовление одинаковых универсальных станков и одинаковых ЭВМ намного дешевле изготовления сложных специализированных станков малыми партиями.

Собрав вместе несколько универсальных станков, складских, транспортных и ремонтных роботов (также работающих под управлением ЭВМ), мы получим так называемое ***гибкое автоматизированное производство***, безлюдный завод-автомат. Человек может перенастроить этот завод на изготовление новой продукции, даже не появляясь на его территории,— достаточно по линиям связи переслать новые программы на ЭВМ завода.

Таким образом, с появлением ЭВМ материальное производство новых станков постепенно «вытесняется» производством прог-

рамм, т. е. производством информационным. Конечно, никакая программа в памяти ЭВМ не в состоянии заменить технологию резания на штамповку или литье. Однако в рамках данной технологии, в рамках тех элементарных операций, которые умеет выполнять данный станок или данный завод-автомат, любые детали могут быть произведены с помощью подходящих программ, помещенных в память ЭВМ. Другими словами, если на ранних стадиях автоматизации производства материальная и информационная составляющие были слиты и для автоматизированного изготовления новых деталей требовались новые станки, то теперь материальная составляющая (т. е. элементарные технологические воздействия: резание, штамповка и пр.) отделяется от информационной (т. е. от того, какую именно деталь надо изготовить с помощью данной технологии).

3.5. ПРОГРАММИРОВАНИЕ — ВТОРАЯ ГРАМОТНОСТЬ

Процесс выделения информационной составляющей из материального производства происходит практически во всех видах деятельности человека. ЭВМ — универсальная машина для обработки информации. Одна и та же ЭВМ, выполняя разные программы, может управлять разными станками и технологическими линиями, разными роботами и устройствами. Поэтому деятельность по составлению программ, по использованию ЭВМ не является специфической для какой-то одной области, ею пронизывается все современное производство.

С развитием общества ЭВМ (а значит, и программирование) начинают проникать во все сферы жизни, будь то производство, здравоохранение, культура, досуг, быт. Разделение материального и информационного производства становится все глубже, а навыки использования ЭВМ, знание основ информатики, навыки алгоритмизации и программирования приобретают все более фундаментальный характер — становятся таким же элементом человеческой культуры, как умение читать, писать, считать и излагать свои мысли. Вот почему ЭВМ и информатика пришли в школы всего мира.

Инициатором введения информатики в школы СССР был видный советский ученый, академик А. П. Ершов.

УПРАЖНЕНИЯ

Опишите точный план действий, приводящий к решению следующих задач:

1. Волк, коза и капуста. На берегу реки стоит крестьянин с лодкой, а рядом с ним — волк, коза и капуста. Крестьянин должен переправиться сам и перевезти волка, козу и капусту на другой берег. Однако в лодку, кроме крестьянина, помещается либо только волк, либо только коза, либо только капуста. Оставлять же волка с козой или козу с капустой без присмотра нельзя — волк может съесть козу, а коза — капусту. Как должен вести себя крестьянин?

2. Два встречных поезда, в каждом из которых паровоз и 21 вагон, встретились на дороге с одним тупиком (рис. 2). Тупик вмещает 11 вагонов или 10 вагонов и паровоз. Как поездам разъехаться (т. е. как должны маневрировать машинисты, чтобы каждый поезд продолжил движение в своем направлении)?

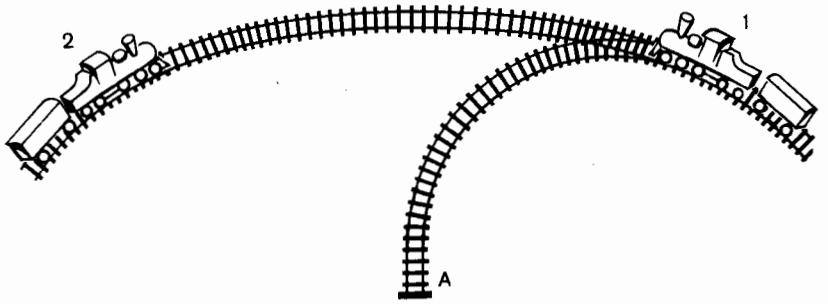


Рис. 2

3. Ханойская башня. На подставке укреплены три стержня, на левый стержень нанизано несколько колец уменьшающегося размера — внизу самое большое кольцо, на нем поменьше, сверху еще меньше и т. п. (рис. 3).

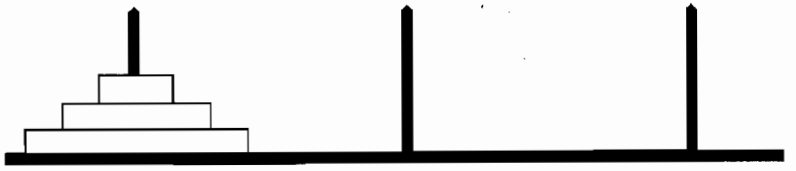


Рис. 3

Надо, перемещая по одному кольцу со стержня на стержень, переместить все кольца на правый стержень, но при этом ни в какой момент времени большее кольцо на меньшее класть нельзя. Опишите, как надо перекладывать кольца, если в начальный момент на левом стержне: а) 1; б) 2; в) 3; г) 4; д*) 64 кольца. (По преданию перекладыванием 64 колец занимаются монахи в одном из буддийских монастырей. Согласно легенде, в момент, когда они кончат перекладывать кольца, наступит конец света. Прикиньте приблизительно, когда это произойдет, если считать, что монахи перекладывают примерно 1 кольцо в секунду.)

4. Фирма "Электронные Приборы" выпустила автоматизированную ванну "Банный Комплекс — 10", управляемую с помощью 10 кнопок "долить 1 л", "долить 2 л", ..., "долить 5 л"; "слить 1 л", "слить 2 л", ..., "слить 5 л", при нажатии на которые доливается или сливается указанное количество литров воды. Однако в результате ошибки фирмы все кнопки, кроме "долить 5 л" и "слить 3 л", не работают. Как долить в ванну 3 л воды? Сколько воды при этом пропадет впустую из-за брака фирмы?

5. Два солдата подошли к реке, по которой на лодке катаются двое мальчиков. Как солдатам переправиться на другой берег, если лодка вмещает только одного солдата либо двух мальчиков, а солдата и мальчика уже не вмещает?

6. Петя и Коля играют в следующую игру: на стол кладется 15 спичек. Ребята по очереди берут их со стола, причем за один ход разрешается взять 1, 2 или 3 спички. Выигрывает тот, кто возьмет последнюю спичку. Первым ходит Петя. Как он должен играть, чтобы выиграть?

7. Есть двое песочных часов: на 3 минуты и на 8 минут. Для приготовления эликсира бессмертия его надо варить ровно 7 минут. Как это сделать?

8. Автоматическое устройство имеет 2 кнопки и экран. При включении на экране загорается число 0. При нажатии на одну кнопку число на экране удваивается (вместо x появляется $2x$). При нажатии на другую кнопку число увеличивается на 1 (вместо x появляется $x+1$). Как надо нажимать на кнопки, чтобы на экране появилось

а) число 5;
б) число 99;
в) число 99, если разрешается нажимать на кнопки не более 10 раз?

9. Придумайте способ нахождения самой легкой и самой тяжелой из 100 монет различной массы, если можно сделать не более 150 взвешиваний на чашечных весах без гирь.

10. Имеется а) 3, б) 4, в) 5, г) 6 монет, среди которых одна фальшивая (легче других). Придумайте способ нахождения фальшивой монеты за минимальное число взвешиваний на чашечных весах без гирь.

11. Имеется 1000 монет, из которых одна фальшивая (легче других). Придумайте способ нахождения фальшивой монеты за 7 взвешиваний на чашечных весах без гирь. Докажите, что нельзя придумать способ, который гарантирует нахождение фальшивой монеты за 6 взвешиваний.

12***. Среди $2n+1$ различных по массе монет нужно найти среднюю (т. е. такую, которая тяжелее n монет и легче других n монет). Придумайте, как это сделать не более чем за $100n$ взвешиваний на чашечных весах без гирь.

ГЛАВА I. АЛГОРИТМИЧЕСКИЙ ЯЗЫК

§ 4. ИСПОЛНИТЕЛЬ «РОБОТ». ПОНЯТИЕ АЛГОРИТМА

4.1. ШКОЛЬНЫЙ АЛГОРИТМИЧЕСКИЙ ЯЗЫК

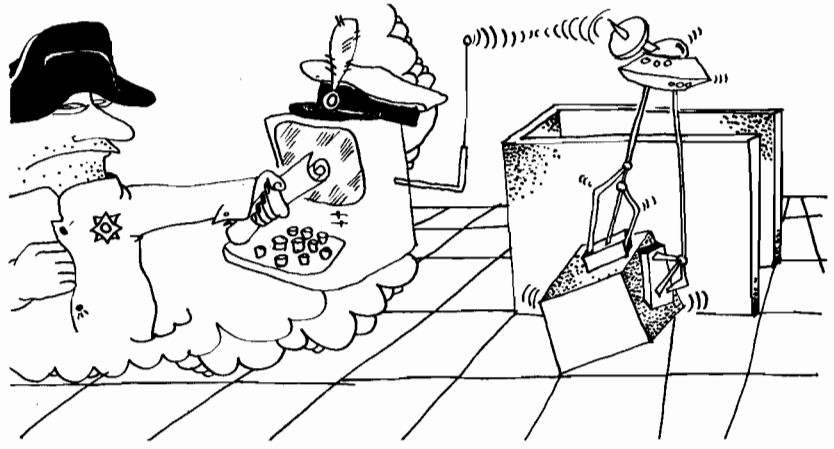
Существует много разных алгоритмических языков. Фортран, Алгол, Паскаль, Бейсик, Ада, Си, Лого, Лисп, Пролог — вот далеко не полный перечень наиболее популярных из них.

Мы будем использовать специальный ***школьный алгоритмический язык.*** Программы на этом языке называются ***алгоритмами*** — их можно выполнить на любой серийной советской школьной ЭВМ. В учебнике мы будем иметь дело только с одним языком и называть его "алгоритмический язык", опуская слово "школьный".

Знакомство с правилами составления и записи алгоритмов на алгоритмическом языке мы начнем с алгоритмов управления исполнителями "Робот" и "Чертежник".

4.2. ИСПОЛНИТЕЛЬ «РОБОТ»

Робот работает на клетчатом "поле" (между клетками могут быть расположены стены) и умещается целиком в одной клетке.



На бумаге, на классной доске и на экране ЭВМ мы будем изображать поле Робота так, как показано на рисунке 4.

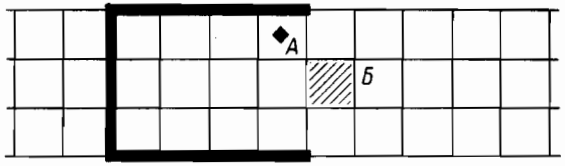


Рис. 4

Робот умеет выполнять всего 17 команд. Мы пока изучим 5: "вверх", "вниз", "вправо", "влево", "закрасить".

По командам "вверх", "вниз", "вправо", "влево" Робот перемещается в соседнюю клетку. В некоторых случаях эти команды могут быть невыполнимы: например, на рисунке 4 выше Робота стена, поэтому команду "вверх" выполнить нельзя.

По команде "закрасить" Робот закрашивает клетку, в которой стоит. Если клетка уже была закрашена, то она будет закрашена еще раз, т. е. останется закрашенной.

4.3. ПРОСТЕЙШИЙ ПРИМЕР АЛГОРИТМА

Пусть требуется перевести Робота из клетки А в клетку Б (рис. 4). При управлении "вручную" мы можем два раза скомандовать Роботу "вправо" и один раз "вниз". Однако, если мы хотим, чтобы Роботом управляла ЭВМ, мы должны записать эти команды в виде алгоритма на алгоритмическом языке:

```
алг ход конем                                   (А1)
 дано |Робот в клетке А (рис. 4)
 надо |Робот в клетке Б (рис. 4)
нач
| вправо
| вправо
| вниз
кон
```

4.4. ОБЩИЙ ВИД АЛГОРИТМА

В простейшем случае алгоритм на алгоритмическом языке записывается так:

```
алг имя алгоритма
 дано условия применимости алгоритма
 надо цель выполнения алгоритма
нач
| тело алгоритма (последовательность команд)
кон
```


Слова **алг** (алгоритм), **дано**, **надо**, **нач** (начало), **кон** (конец) называются ***служебными словами*** и служат для оформления алгоритма. Часть алгоритма до служебного слова **нач** называют ***заголовком*** алгоритма, а часть между словами **нач** и **кон** — ***телом*** алгоритма. Имя (название) алгоритма может быть любым. Обычно оно подбирается так, чтобы можно было понять, для чего предназначен алгоритм. Служебные слова **алг**, **нач** и **кон** пишутся строго одно под другим, **нач** и **кон** соединяются вертикальной чертой, правее которой пишется тело алгоритма — последовательность команд.

4.5. КОММЕНТАРИИ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

В алгоритме "ход конем" (А1) после знака | в строках **дано** и **надо** записан комментарий. Такие комментарии разрешается помещать в конце любой строки, отделяя их знаком |. Если комментарий занимает несколько строк, то знак | перед комментарием надо писать в каждой строке. Комментарии могут записываться в любой удобной для человека форме. При выполнении алгоритма ЭВМ полностью пропускает комментарии — алгоритм выполняется так же, как если бы комментариев вообще не было. Таким образом, комментарии предназначены исключительно для человека — они облегчают понимание алгоритма.

4.6. ВЫЗОВ КОМАНДЫ ИСПОЛНИТЕЛЯ

При выполнении алгоритма ЭВМ последовательно командует Роботу выполнять записанные в алгоритме команды. Каждый такой приказ на выполнение команды называется ***вызовом*** этой команды.

4.7. ОШИБКИ В АЛГОРИТМАХ

Если при составлении алгоритма мы случайно вместо "вниз" напишем "внис" или вместо "вправо" напишем "направо", то ЭВМ нашу запись не поймет и, даже не приступая к выполнению алгоритма, сообщит об ошибке. Такие ошибки в записи алгоритма называются ***синтаксическими***.

Ошибки другого вида — ***отказы*** — проявляются при выполнении алгоритма. Например, при попытке выполнить алгоритм "ход конем" (А1) в обстановке, изображенной на рисунке 5, ЭВМ попытается последовательно вызвать команды "вправо", "вправо",

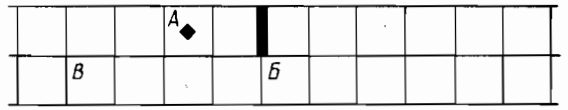


Рис. 5

"вниз". Однако вторую команду "вправо" Робот выполнить не сможет. Такие ситуации называются отказами. Если при выполнении алгоритма возникнет отказ, то ЭВМ сообщит об ошибке и прекратит выполнение.

Кроме синтаксических ошибок и отказов, в алгоритме могут быть ***логические*** ошибки, не обнаруживаемые ЭВМ ни до выполнения алгоритма, ни при его выполнении. Например, если в алгоритме А1 мы вместо "вправо" случайно напишем "влево", то ЭВМ выполнит алгоритм, Робот из клетки А (рис. 5) переместится в клетку В (рис. 5), но никаких сообщений об ошибках мы не получим (да и откуда ЭВМ знать, куда мы на самом деле хотели переместить Робота!).

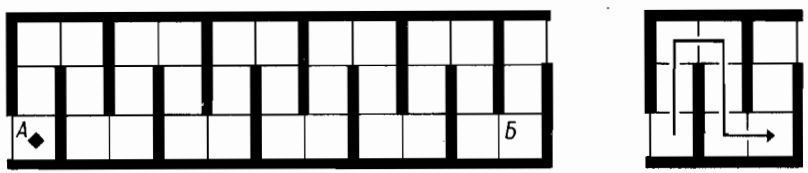
В правильно составленных алгоритмах никаких ошибок быть не должно. Но если синтаксические ошибки и отказы обычно легко устранимы, то поиск и устранение логических ошибок может оказаться весьма трудным делом.

4.8. ЗАПИСЬ НЕСКОЛЬКИХ КОМАНД В ОДНОЙ СТРОКЕ

Правила алгоритмического языка разрешают записывать в одной строке несколько команд через точку с запятой. Пусть требуется перевести Робота из клетки А в клетку Б (рис. 6, *а*).

Путь, который должен пройти Робот, можно разбить на пять одинаковых участков (рис. 6, *б*). Команды прохождения каждого участка можно сгруппировать в одну строку — это сокращает запись алгоритма и делает его более понятным:

алг из А в Б (А2)
дано | Робот в клетке А (рис. 6, *а*)
надо | Робот в клетке Б (рис. 6, *а*)
нач
вверх; вверх; вправо; вниз; вниз; вправо
вверх; вверх; вправо; вниз; вниз; вправо
вверх; вверх; вправо; вниз; вниз; вправо
вверх; вверх; вправо; вниз; вниз; вправо
вверх; вверх; вправо; вниз; вниз; вправо
кон



а) *б)*

Рис. 6

УПРАЖНЕНИЯ

1. Дан алгоритм, в котором стерты комментарии и название:

а) **алг** (А3)
дано |
надо |
нач
вверх; закрасить; вниз
вправо; закрасить; влево
вниз; закрасить; вверх
влево; закрасить; вправо
кон

б) **алг** (А4)
дано |
надо |
нач
вверх; вправо; закрасить
вниз; вниз; закрасить
влево; влево; закрасить
вверх; вверх; закрасить
вправо; вниз
кон

Опишите движение Робота в процессе выполнения алгоритма. Нарисуйте конечное положение Робота и закрашенные в результате выполнения клетки. Придумайте подходящее название алгоритма и впишите комментарии после слов **дано** и **надо**.

2. Измените алгоритм задачи 1, б) так, чтобы при его исполнении Робот:
а) прошел тем же маршрутом, но ничего не закрашивал;
б) закрасил все клетки, в которых он побывал.

3. Известно, что на поле Робота нет стен и закрашенных клеток. Определите, сколько клеток будет закрашено после исполнения следующих команд:

а)
закрасить
вправо
вверх
закрасить
вправо
закрасить
вверх
закрасить
закрасить
вправо

б)
закрасить
вправо
закрасить
закрасить
вправо
вправо
закрасить
закрасить
закрасить
вправо

4. Известно, что на поле Робота имеется одна стена, равная по длине стороне клетки, а попытка выполнить последовательность команд из задачи 3, а) приводит к отказу. Измените последовательность команд так, чтобы конечное положение Робота осталось прежним, закрашивались те же клетки, а отказа не возникало.

5. Составьте алгоритм, при выполнении которого Робот переместится из клетки А в клетку Б (рис. 7).

6. Приведите все алгоритмы из трех команд, переводящие Робота из клетки А в клетку Б в задаче 5, а.

7. Существует ли алгоритм для задачи 5, а, при выполнении которого Робот делает: а) два шага; б) четыре шага; в) семь шагов; г) 1990 шагов?

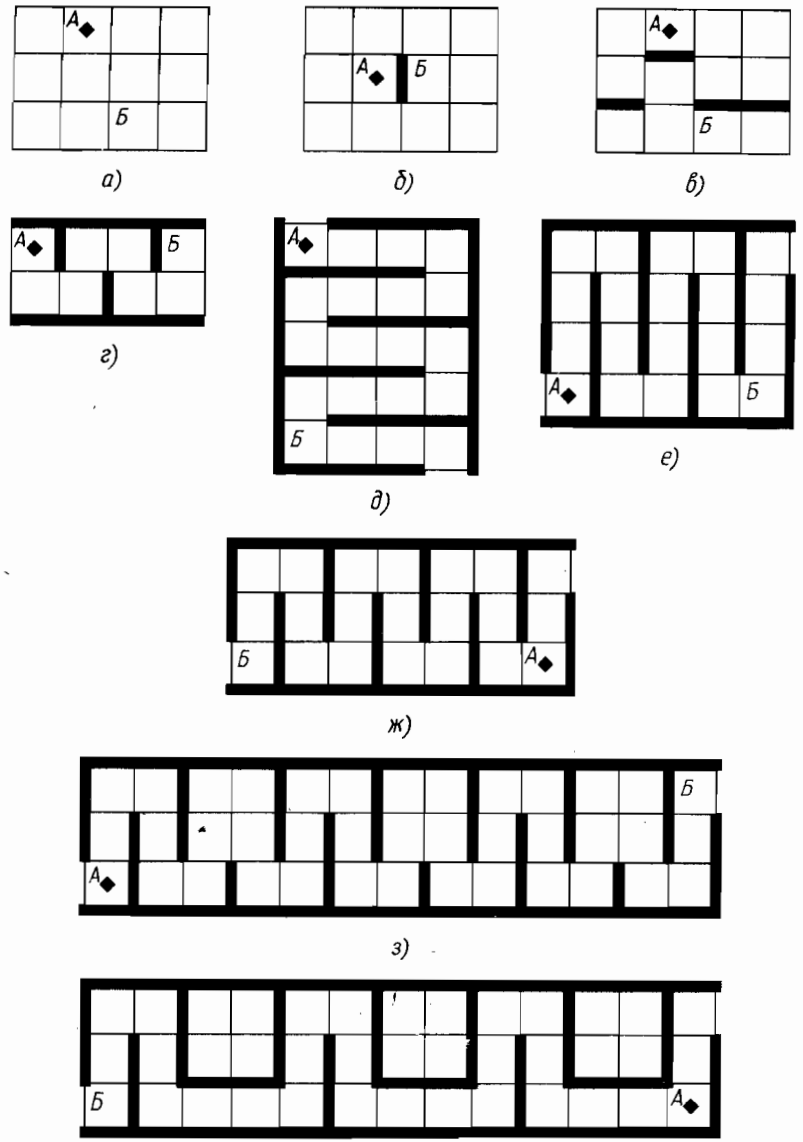


Рис. 7

8. Составьте алгоритм, который перемещает Робота из А в Б и закрашивает клетки, отмеченные точками (рис. 8).

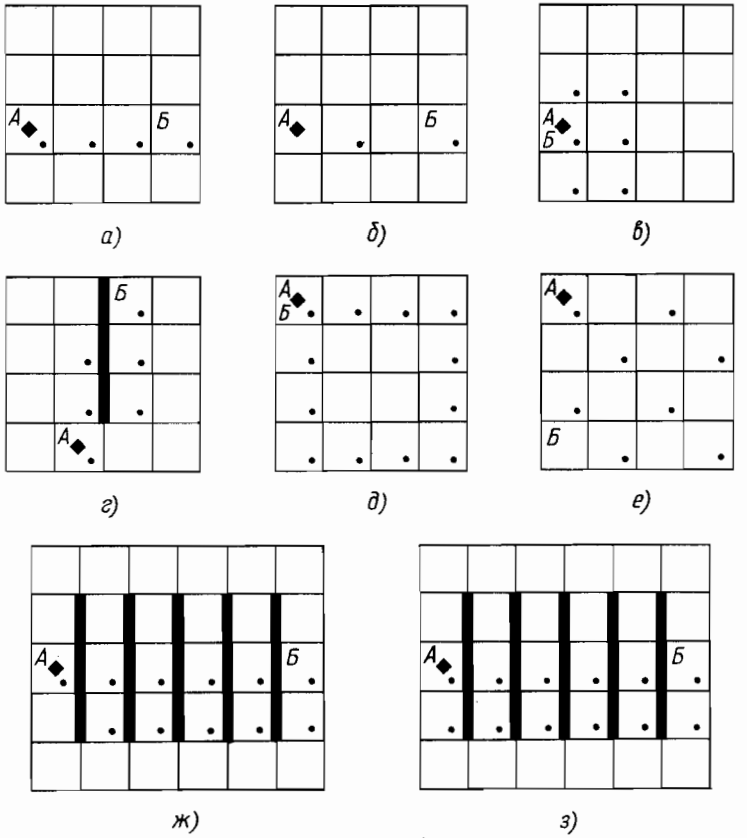


Рис. 8

7. Петя составил алгоритм, а Коля стер в нем одну команду:

алг прогулка (А5)
дано | стен на поле нет
надо | Робот погулял и вернулся в исходное положение
нач
| вверх
| вправо
| ???
| вниз
| влево
| влево
кон

Определите, что за команду стер Коля, если известно, что при выполнении составленного Петей алгоритма Робот возвращался в исходное положение.

8. Петя составил алгоритм, при выполнении которого Робот вернулся в исходное положение. Коля стер одну из команд. При выполнении Колиного алгоритма Робот также вернулся в исходное положение. Какую команду стер Коля?

9. Петя составил алгоритм, при выполнении которого на поле без стен Робот вернулся в исходное положение. Коля переставил две команды местами. Докажите, что при выполнении Колиного алгоритма Робот также вернется в исходное положение.

10. Петя составил алгоритм для Робота, который на поле без стен и закрашенных клеток закрашивает 5 клеток. Коля переставил в алгоритме две соседние команды. Может ли новый алгоритм закрашивать а) 3; б) 4; в) 5; г) 6; д) 7 клеток?

11. Петя составил алгоритм, при выполнении которого Робот закрашивает 5 клеток. Коля переставил в алгоритме какие-то две команды (необязательно соседние). Может ли новый алгоритм закрашивать а) 0; б) 1; в) 5; г) 7; д) 100 клеток?

12. Петя составил алгоритм, переводящий Робота из клетки А в клетку Б с закрашиванием каких-то клеток. Что должен сделать Коля с этим алгоритмом, чтобы получить алгоритм, переводящий Робота из Б в А и закрашивающий те же клетки?

§ 5. ИСПОЛНИТЕЛЬ «ЧЕРТЕЖНИК» И РАБОТА С НИМ

5.1. ОСОБЕННОСТИ ЗАПИСИ ЧИСЕЛ В ИНФОРМАТИКЕ

В информатике для отделения целой части числа от дробной используется точка, а не запятая. Это позволяет записывать несколько рядом стоящих чисел через запятую без риска вызвать путаницу. При задании точек плоскости координаты x и y в информатике разделяются запятой (рис. 9).

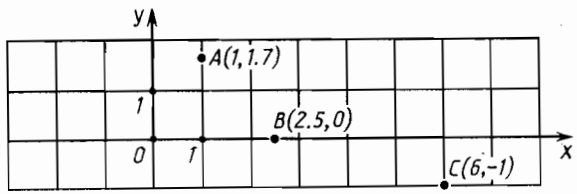


Рис. 9

5.2. ИСПОЛНИТЕЛЬ «ЧЕРТЕЖНИК»

Чертежник предназначен для построения рисунков, чертежей, графиков и т. д. на бесконечном листе бумаги. Чертежник имеет перо, которое можно поднимать, опускать и перемещать.

При перемещении опущенного пера за ним остается след — отрезок от старого положения пера до нового. Всего Чертежник умеет выполнять 4 команды:

поднять перо
опустить перо
сместиться в точку (**арг** **вещ** x, y)
сместиться на вектор (**арг** **вещ** a, b)

Слова **арг** ***(аргументы)*** и **вещ** ***(вещественные)*** указывают, что команды "сместиться в точку" и "сместиться на вектор" имеют аргументы, которые могут быть произвольными вещественными числами. Аргументы указываются при вызове команды, например: "сместиться в точку (2.5,0.5)".

5.3. РАБОТА КОМАНДЫ «СМЕСТИТЬСЯ НА ВЕКТОР»

Если перо Чертежника находится в точке (x, y), то по команде "сместиться на вектор (a, b)" Чертежник сместит перо в точку с координатами (x+a, y+b) (рис. 10,*а*). Если перо опущено, то при этом будет нарисован отрезок от старого положения пера до нового — от точки (x, y) до точки (x+a, y+b). Этот отрезок является изображением вектора с координатами (a, b) и началом в точке (x, y), чем и объясняется название команды.

Если, например, перо Чертежника опущено и находится в точке (1, 2), то при выполнении команды "сместиться на вектор (3, 3)" Чертежник сместит перо в точку (1+3, 2+3), т. е. нарисует отрезок от точки (1, 2) до точки (4, 5) (рис. 10, *б*).

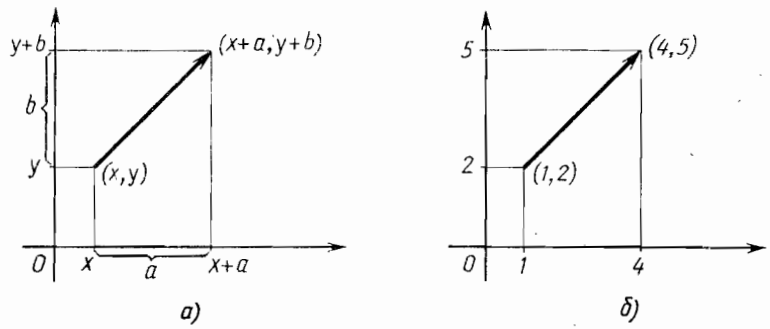


Рис. 10

5.4. ПРИМЕР АЛГОРИТМА УПРАВЛЕНИЯ ЧЕРТЕЖНИКОМ

Попробуем с помощью Чертежника нарисовать прямоугольник с двумя диагоналями (рис. 11, *а*). Как это сделать? Можно начать с левого нижнего угла и, двигаясь против часовой стрелки,

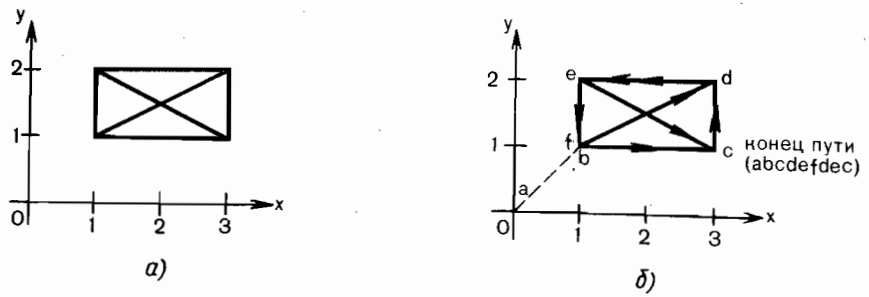


Рис. 11

нарисовать все 4 стороны прямоугольника, не отрывая пера от бумаги, после чего нарисовать диагонали (рис. 11, *б*).

Какие же команды и в какой последовательности нужно для этого дать Чертежнику? Пусть в начальный момент перо поднято. В левый нижний угол можно попасть, скомандовав

сместиться в точку (1, 1)

Теперь надо нарисовать прямоугольник. Для этого прежде всего следует вызвать команду:

опустить перо

После этого перо будет расположено в точке (1, 1) и опущено. Для рисования прямоугольника воспользуемся командой "сместиться в точку":

сместиться в точку (3, 1)
сместиться в точку (3, 2)
сместиться в точку (1, 2)
сместиться в точку (1, 1)

После выполнения этих команд прямоугольник будет нарисован, а опущенное перо будет расположено снова в левом нижнем углу — в точке (1, 1).

Осталось нарисовать диагонали. Это можно сделать так:

сместиться в точку (3, 2)
сместиться в точку (1, 2)
сместиться в точку (3, 1)

При этом, однако, верхняя сторона прямоугольника будет нарисована второй раз. Если мы хотим этого избежать, то перед командой

сместиться в точку (1, 2)

нужно поднять перо, а потом его опустить:

сместиться в точку (3, 2)
поднять перо
сместиться в точку (1, 2)
опустить перо
сместиться в точку (3, 1)

В этот момент вся картинка изображена и осталось поднять перо, чтобы лист бумаги можно было вынуть:

поднять перо

Запишем полученный алгоритм на алгоритмическом языке:

алг прямоугольник с диагоналями (А6)
дано | перо поднято
надо | нарисован прямоугольник с диагоналями (рис. 11, *а*),
| перо поднято
нач
| сместиться в точку (1, 1)
| опустить перо
| сместиться в точку (3, 1)
| сместиться в точку (3, 2)
| сместиться в точку (1, 2)
| сместиться в точку (1, 1)
| | нарисован прямоугольник
| сместиться в точку (3, 2)
| поднять перо
| сместиться в точку (1, 2)
| опустить перо
| сместиться в точку (3, 1)
| поднять перо
кон

5.5. РИСОВАНИЕ БУКВ

С помощью Чертежника можно рисовать любые фигуры, составленные из отрезков, например буквы. Составим алгоритм, при выполнении которого Чертежник рисует на клетчатой бумаге букву М (рис. 12; размеры каждой клетки 1×1). Поскольку начальное положение пера на плоскости не задано, то придется воспользоваться командой "сместиться на вектор":

алг буква М (А7)
дано | перо в точке А (рис. 12) и поднято
надо | нарисована буква М (рис. 12),
| перо в точке Б и поднято
нач
| опустить перо
| сместиться на вектор (0, 4)
| сместиться на вектор (1, −2)
| сместиться на вектор (1, 2)
| сместиться на вектор (0, −4)
| поднять перо
| сместиться на вектор (1, 0)
кон

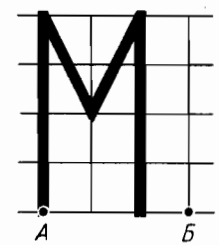


Рис. 12

Аналогично можно составить алгоритмы "буква И", "буква Р" и "буква У" (рис. 13).

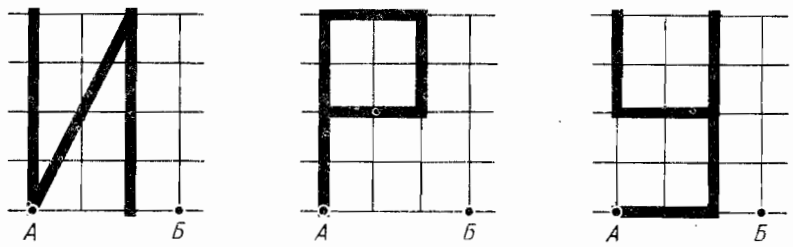


Рис. 13

5.6. ПОСЛЕДОВАТЕЛЬНОЕ ВЫПОЛНЕНИЕ АЛГОРИТМОВ

Пусть перо Чертежника расположено в точке (0, 0) и мы последовательно приказываем ЭВМ выполнять алгоритмы "буква М", "буква И", "буква Р". Что будет нарисовано в результате?

Перед выполнением алгоритма "буква М" перо находится в точке (0, 0). После выполнения перо окажется в точке (3, 0), с которой и начнет выполняться алгоритм "буква И". После его выполнения перо окажется в точке (6, 0). Именно с этой точки будет нарисована буква Р. В итоге будет написано слово МИР (рис. 14), а перо окажется в точке (9, 0).

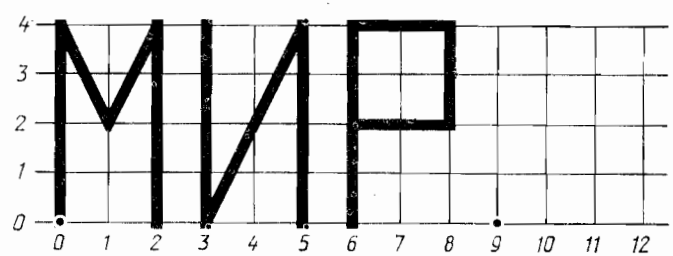


Рис. 14

УПРАЖНЕНИЯ

1. Петя записал через запятую несколько вещественных и целых чисел, по привычке поставив десятичные запятые внутри чисел. Вот что у него получилось:

а) 3, 5, 7; б) 7, 3, 5, 0, 1.

Сколькими способами можно прочесть эти записи, если в а) записано два числа, а количество чисел, записанных в б), неизвестно?

2. Нарисуйте результат выполнения следующего алгоритма:

(А8)

```
алг домик
 дано |перо поднято
 надо |нарисован домик, перо в исходном положении и поднято
нач
| опустить перо
| сместиться на вектор (4, 0)
| сместиться на вектор (0, 4)
| сместиться на вектор (—4, 0)
| сместиться на вектор (0, —4)
| поднять перо
| сместиться на вектор (0, 4)
| опустить перо
| сместиться на вектор (2, 2)
| сместиться на вектор (2, —2)
| поднять перо
| сместиться на вектор (—4, —4)
кон
```

3. Измените алгоритм "домик" (А8) так, чтобы домик рисовался с окошком.

4. Составьте алгоритм управления Чертежником, после выполнения которого будут нарисованы:

а) отрезок с концами в точках (1, 2) и (—1, 1);

б) квадрат со сторонами длины 4, параллельными координатным осям, так, чтобы левый нижний угол квадрата совпадал с начальным положением пера Чертежника;

в) квадрат со сторонами длины 6, параллельными координатным осям, так, чтобы левый нижний угол квадрата совпадал с начальным положением пера Чертежника;

г) какой-нибудь отрезок длины 3, проходящий через точку (2, 2);

д) какой-нибудь квадрат с сторонами длины 2 и центром в начале координат;

е) какой-нибудь прямоугольник с длинами сторон 3 и 4, содержащий внутри себя начало координат;

ж) какой-нибудь параллелограмм.

5. Составьте алгоритм управления Чертежником, после исполнения которого будут нарисованы:

а) инициалы полководца Кутузова;

б) ваши инициалы;

в) буква Ф;

г) зеркальные отражения букв И и Р относительно горизонтальной оси;

д) число 12 римскими цифрами;

е) слово "МГУ";

ж) слово "СССР";

з) почтовый индекс 161110 (рис. 15);

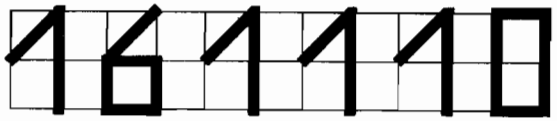


Рис. 15

и) ваш почтовый индекс.

6. Составьте алгоритм управления Чертежником, после исполнения которого будут нарисованы:

а) какой-нибудь треугольник, у которого одна из сторон больше 1;

б) какой-нибудь треугольник и его три медианы;

в) прямоугольный треугольник, у которого катеты не параллельны осям координат;

г) множество точек, удовлетворяющих условию $|x|+|y|=1$.

7. Составьте алгоритм управления Чертежником, после выполнения которого будет нарисован график функции $y=f(x)$ на отрезке $[-4, 4]$, где

а) $f(x)=|x|$,
б) $f(x)=|x|-2$,
в) $f(x)=||x|-2|$.

8. Составьте десять алгоритмов для рисования десяти цифр почтового индекса так, чтобы при их последовательном выполнении цифры рисовались друг за другом (как в упражнении 5, *з*).

9. Составьте алгоритм для рисования фигур, изображенных на рисунке 16, так, чтобы в процессе рисования перо не отрывалось от бумаги и ни одна линия не проводилась дважды.

а) *раскрытый конверт*

б) *звезда*

в) *шаблон цифры почтового индекса*

Рис. 16

10. Измените алгоритмы рисования букв М, И, Р так, чтобы при последовательном выполнении этих алгоритмов слово МИР оказалось написанным:

а) с удвоенным расстоянием между буквами;

б) буквами удвоенного размера;

в) сверху вниз (рис. 17);

г) сверху вниз буквами удвоенного размера.

11. Дан алгоритм:

алг фигура (А9)
дано | перо в начале координат и поднято
нач
| сместиться в точку (2, 1)
| опустить перо
| сместиться на вектор (0, 3)
| сместиться на вектор (1, 0)
| сместиться на вектор (0, −1)
| сместиться на вектор (1, 0)
| сместиться на вектор (0, −1)
| сместиться на вектор (1, 0)
| сместиться на вектор (0, −1)
| сместиться в точку (2, 1)
| поднять перо
кон

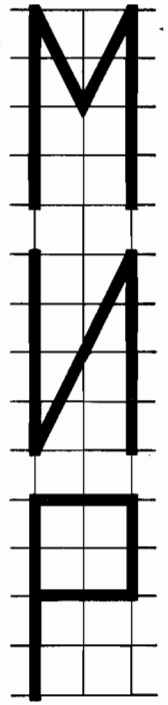


Рис. 17

а) не выполняя алгоритма и не рисуя получившейся фигуры, определите, где будет расположено перо после выполнения алгоритма, будет ли оно поднято или опущено;

б) выполните алгоритм, нарисуйте получившуюся фигуру;

в) переделайте алгоритм так, чтобы он рисовал где-нибудь на плоскости фигуру вдвое большего размера;

г) переделайте алгоритм так, чтобы он рисовал фигуру, симметричную первой относительно оси y;

д) определите, что будет нарисовано, если в алгоритме изменить знаки всех аргументов на противоположные.

12. Дан алгоритм:

алг ломаная (А10)
дано | перо в начале координат и поднято
нач
| опустить перо
| сместиться на вектор (1, 3)
| сместиться на вектор (1, 2)
| сместиться на вектор (1, 1)
| сместиться на вектор (1, 0)
| сместиться на вектор (1, −1)
| сместиться на вектор (1, −2)
| сместиться на вектор (1, −3)
| поднять перо
кон

Не выполняя алгоритма и не рисуя получившейся ломаной, определите:

а) будет перо после выполнения поднято или опущено;
б) координату x конечного положения пера;
в) координату y конечного положения пера;
г) будет ли ломаная замкнута;
д) расстояние между концами ломаной.

Нарисуйте ломаную, проверьте ваши ответы.

13. Спроектируйте систему команд автоматического устройства, которое рисует разными цветами. Составьте несколько алгоритмов для управления этим устройством и нарисуйте результаты их выполнения.

14. Придумайте автоматическое устройство, которое умеет устанавливать, перемещать и убирать стены на поле Робота. Придумайте несколько задач по управлению этим устройством.

§ 6. ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ. АЛГОРИТМЫ С АРГУМЕНТАМИ

6.1. АЛГОРИТМ РИСОВАНИЯ СЛОВА МИР

Составив три алгоритма для рисования букв М, И, Р, можно не только последовательно приказывать ЭВМ выполнять эти алгоритмы, но и задать порядок их выполнения в виде алгоритма:

```
алг МИР                                              (А11)
 дано |перо поднято
 надо |нарисовано слово МИР, перо поднято и расположено
      |в конце слова (начале следующей буквы)
нач
| буква М
| буква И
| буква Р
кон
```

Строки ''буква М'', ''буква И'', ''буква Р'' в записи алгоритма ''МИР'' (А11) — это команды алгоритмического языка, которые означают соответственно

выполнить алгоритм с именем ''буква М'',
выполнить алгоритм с именем ''буква И'',
выполнить алгоритм с именем ''буква Р''.

Если теперь поместить в ЭВМ алгоритм ''МИР'' и алгоритмы ''буква М'', ''буква И'', ''буква Р'' (см. А11), а затем приказать ЭВМ выполнить алгоритм ''МИР'', то ЭВМ автоматически выполнит сначала алгоритм ''буква М'', потом — ''буква И'' и наконец — ''буква Р''.

6.2. ПОНЯТИЯ ОСНОВНОГО И ВСПОМОГАТЕЛЬНОГО АЛГОРИТМОВ. ВЫЗОВ ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА

Алгоритмы ''буква М'', ''буква И'' и ''буква Р'' в этом примере называются вспомогательными для основного алгоритма ''МИР''.

Приказ на выполнение вспомогательного алгоритма называется ***вызовом*** этого вспомогательного алгоритма.

В общем случае если в записи алгоритма X встречается вызов алгоритма Y, то алгоритм Y называется ***вспомогательным*** для X, а алгоритм X называется ***основным*** для Y.

6.3. ОДИН И ТОТ ЖЕ АЛГОРИТМ МОЖЕТ ВЫСТУПАТЬ И В РОЛИ ВСПОМОГАТЕЛЬНОГО, И В РОЛИ ОСНОВНОГО

Алгоритм ''МИР'' (А11) сам может быть использован как вспомогательный. Например, мы можем составить следующий алгоритм для рисования лозунга ''МИРУ МИР'':

```
алг МИРУ МИР                                         (А12)
 дано |перо поднято
 надо |нарисован лозунг ''МИРУ МИР'', перо поднято и распо-
      |ложено в конце лозунга (в начале следующей буквы)
нач
| МИР
| буква У
| сместиться на вектор (3, 0)   |пропуск между словами
| МИР
кон
```

В этом примере алгоритм ''МИР'' является вспомогательным по отношению к алгоритму ''МИРУ МИР'' и основным по отношению к алгоритму ''буква М''.

6.4. ПРИМЕР ИСПОЛЬЗОВАНИЯ ВСПОМОГАТЕЛЬНЫХ АЛГОРИТМОВ

Пусть требуется написать алгоритм, который проводит Робота из клетки А в клетку Б и закрашивает клетки, отмеченные точками (рис. 18). На рисунке показан один из возможных путей Робота при решении этой задачи. Видно, что Робот должен закрасить три прямоугольных ''блока'' размером 3×5 клеток, а между закрашиваниями блоков два раза обойти стену. Запишем эту мысль в виде основного алгоритма:

```
алг из А в Б с закрашиванием                         (А13)
 дано |Робот в точке А (рис. 18)
 надо |Робот в точке Б (рис. 18), нужные клетки закрашены
```

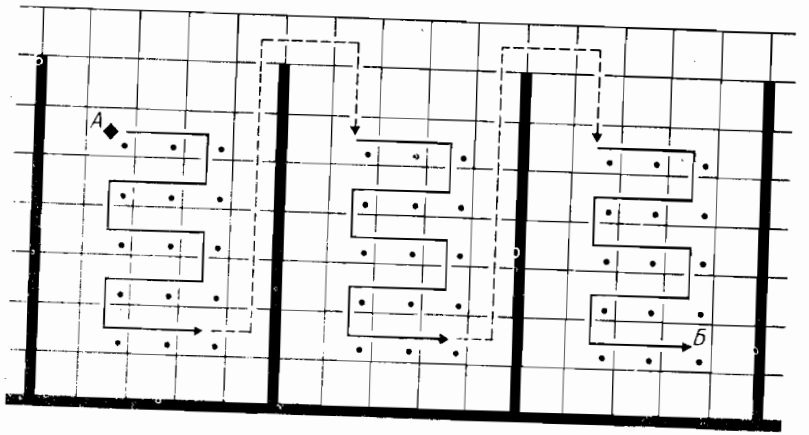


Рис. 18

```
нач
| закрашивание блока
| обход стены
| закрашивание блока
| обход стены
| закрашивание блока
кон
```

Теперь составим вспомогательные алгоритмы, используемые в основном алгоритме "из А в Б с закрашиванием" (А13):

```
алг закрашивание блока                                   (А14)
 дано | Робот в левом верхнем углу блока 3×5 клеток
 надо | блок закрашен, Робот в его правом нижнем углу
нач
| закрасить; вправо; закрасить; вправо; закрасить; вниз
| закрасить; влево;  закрасить; влево;  закрасить; вниз
| закрасить; вправо; закрасить; вправо; закрасить; вниз
| закрасить; влево;  закрасить; влево;  закрасить; вниз
| закрасить; вправо; закрасить; вправо; закрасить
кон
```

```
алг обход стены                                          (А15)
 дано | Робот в правом нижнем углу блока (рис. 18)
 надо | Робот в левом верхнем углу следующего блока (рис. 18)
нач
| вправо; вверх; вверх; вверх; вверх; вверх; вверх
| вправо; вправо; вниз; вниз
кон
```

6.5. МЕТОД ПОСЛЕДОВАТЕЛЬНОГО УТОЧНЕНИЯ

В разобранном примере мы сначала написали основной алгоритм "из А в Б с закрашиванием" (А13), используя еще не написанные вспомогательные алгоритмы для крупных действий

"закрашивание блока" и "обход стены", и лишь потом составили эти вспомогательные алгоритмы А14 и А15. Такой метод составления алгоритмов называется ***методом последовательного уточнения.***

В общем случае метод последовательного уточнения состоит в том, что исходная задача разбивается на ряд крупных частей (подзадач) и составляется основной алгоритм, в котором для решения подзадач используются вызовы еще не написанных вспомогательных алгоритмов. Таким образом, сначала полностью записывается основной алгоритм и только потом начинается составление вспомогательных. При составлении вспомогательных алгоритмов в свою очередь могут использоваться вспомогательные алгоритмы для выполнения более мелких действий и т. д.

Метод последовательного уточнения облегчает составление алгоритмов, так как позволяет решать задачу по частям и использовать в качестве вспомогательных алгоритмы еще не решенных задач.

6.6. РАЗДЕЛЕНИЕ ТРУДА МЕЖДУ ЭВМ И ИСПОЛНИТЕЛЯМИ

Алгоритм — это описание последовательности действий ЭВМ. Исполнители ничего ни про какие алгоритмы не знают.

Например, при выполнении алгоритма "из А в Б с закрашиванием" (А13) именно ЭВМ "разбирается" в записи алгоритма на алгоритмическом языке; анализирует заключенную в нем информацию; "понимает", что сначала надо выполнить алгоритм "закрашивание блока" (А14); выполняет его и т. п. Робот же лишь выполняет последовательно поступающие от ЭВМ команды: "закрасить", "вправо", "закрасить", "вправо", "закрасить", "вниз", "закрасить", "влево" и т. д.

6.7. АЛГОРИТМЫ С АРГУМЕНТАМИ

Вспомните упражнения 4б и 4в из предыдущего параграфа — для рисования квадратов с длинами сторон 4 и 6 мы составляли два разных алгоритма. А как быть, если нужно рисовать много разных квадратов с разными длинами сторон? Не писать же для каждого квадрата свой алгоритм! Оказывается, алгоритмы, как и команды исполнителя, могут иметь ***аргументы*** (**арг**). Правила алгоритмического языка позволяют, например, составить следующий алгоритм рисования квадрата с произвольной длиной стороны:

```
алг квадрат (арг вещ а)                                  (А16)
 дано | перо Чертежника в левом нижнем углу А будущего квад-
      | рата и поднято (рис. 19)
 надо | нарисован квадрат с длиной стороны а (рис. 19),
      | перо Чертежника в точке А и поднято
```

```
нач
| опустить перо
| сместиться на вектор (a, 0)
| сместиться на вектор (0, a)
| сместиться на вектор (—a, 0)
| сместиться на вектор (0, —a)
| поднять перо
кон
```

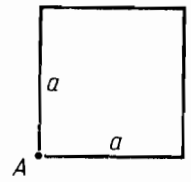


Рис. 19

Запись "**алг** квадрат (**арг** **вещ** а)" означает, что у алгоритма "квадрат" есть один аргумент (**арг**) "а", который может быть произвольным вещественным (**вещ**) числом.

Позже мы познакомимся с алгоритмами, аргументы которых могут быть лишь целыми (**цел**) числами; с алгоритмами, аргументы которых вообще не являются числами; а также с алгоритмами, у которых, кроме аргументов, есть результаты (**рез**). Именно этим объясняется наличие слов **арг** и **вещ** в заголовке алгоритма "квадрат".

6.8. ВЫПОЛНЕНИЕ ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА С АРГУМЕНТАМИ

Если в основном алгоритме написать вызов "квадрат (4)", то ЭВМ запомнит, что аргумент а равен 4, и при выполнении алгоритма "квадрат (**арг** **вещ** а)" скомандует Чертежнику:

```
опустить перо
сместиться на вектор (4, 0)
сместиться на вектор (0, 4)
сместиться на вектор (—4, 0)
сместиться на вектор (0, —4)
поднять перо
```

Если же в основном алгоритме написать вызов "квадрат (6)", то ЭВМ запомнит, что аргумент а равен 6, и скомандует Чертежнику:

```
опустить перо
сместиться на вектор (6, 0)
сместиться на вектор (0, 6)
сместиться на вектор (—6, 0)
сместиться на вектор (0, —6)
поднять перо
```

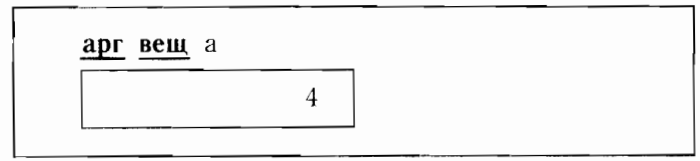
6.9. МОДЕЛЬ ПАМЯТИ ЭВМ

Поясним подробнее, как ЭВМ запоминает значения аргументов в своей памяти. Память ЭВМ удобно представлять в виде обычной классной доски, на которой можно записывать информацию, читать, стирать, записывать заново и т. д. Место, отводимое в памяти ЭВМ для запоминания информации, удобно изображать прямоугольником:

арг **вещ** а

Сама информация (значение аргумента) записывается внутри прямоугольника. Сверху пишется, что это за информация. При выполнении вызова "квадрат (4)" ЭВМ вначале выделит часть памяти для алгоритма "квадрат" (эту часть также удобно изображать в виде прямоугольника) и запомнит значение аргумента:

алг квадрат



При выполнении алгоритма ЭВМ будет подставлять вместо аргумента а его значение 4, а по окончании — сотрет прямоугольник алгоритма "квадрат" со всем его содержимым.

При следующем вызове, например при вызове "квадрат (6)", для алгоритма "квадрат" снова будет выделен прямоугольник в памяти, запомнено новое значение аргумента (6) и т. д.

УПРАЖНЕНИЯ

1. По образцу алгоритма "МИР" (А11) составьте алгоритмы а) РИМ; б) МИМ.

2. Дан основной алгоритм "улица из трех домиков":

```
алг улица из трех домиков                                (А17)
нач
| домик; сместиться на вектор (6, 0)
| домик; сместиться на вектор (6, 0)
| домик
кон
```

Этот алгоритм использует вспомогательный алгоритм "домик" (А8). Нарисуйте результат выполнения алгоритма А17 (полученную картинку и положение пера Чертежника).

3. Составьте алгоритм, рисующий улицу из шести домиков.

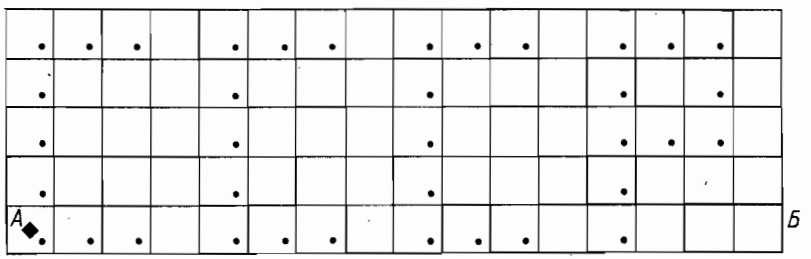
4. Петя зачеркнул последнюю команду "сместиться на вектор (—4, —4)" в алгоритме "домик" (А8). Как Коля должен изменить алгоритм "улица из трех домиков", чтобы рисовалась та же картинка, что и раньше?

5. Составьте алгоритм управления Чертежником, после исполнения которого будет нарисован квадрат 4×4, заштрихованный горизонтальными и (или) вертикальными линиями следующим образом (расстояние между линиями равно 0, 4) (рис. 20).

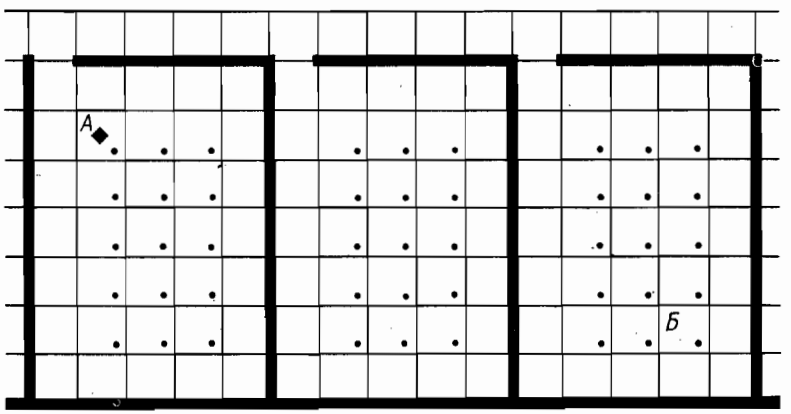
а) *б)* *в)*

Рис. 20

6. Составьте алгоритм, который переводит Робота из А в Б и закрашивает клетки, отмеченные точками (рис. 21, 22).

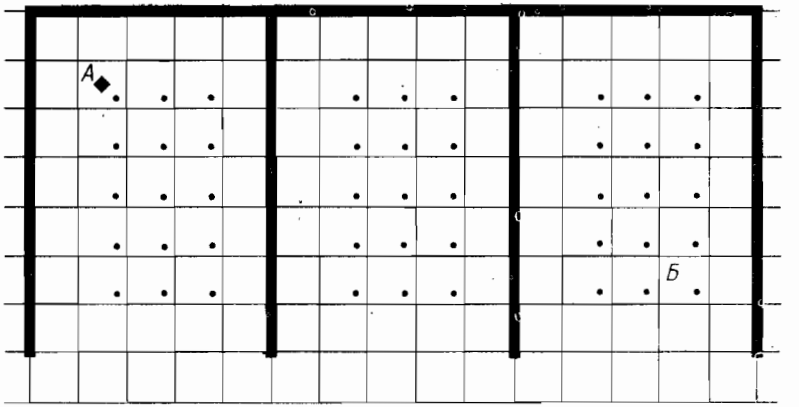


а)

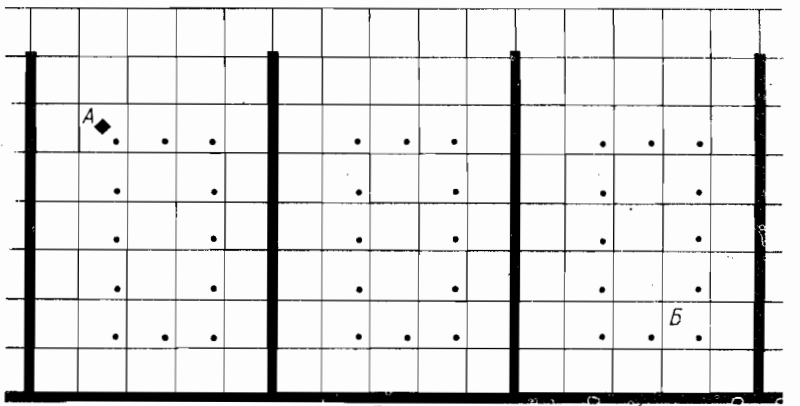


б)

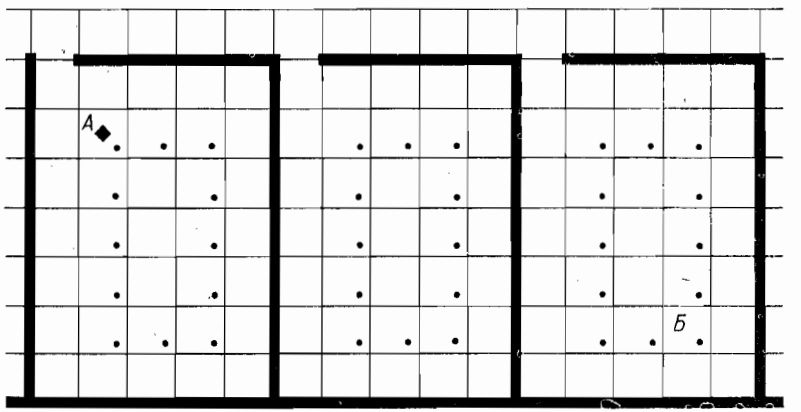
Рис. 21



а)



б)



в)

Рис. 22

7. Даны алгоритмы:

алг фигура (А18)
нач
| фрагмент; фрагмент; фрагмент; фрагмент; фрагмент
кон

алг фрагмент (А19)
нач
| закрасить; вправо; закрасить; вправо; закрасить; вправо
| закрасить; вниз; закрасить; вниз; закрасить; вниз
| закрасить; влево; закрасить; влево; закрасить; влево
| закрасить; вверх; закрасить; вверх; закрасить
| вправо; вправо; вправо; вниз; вниз
кон

Нарисуйте результат выполнения алгоритма "фигура" (закрашенные клетки, начальное и конечное положения Робота).

8. Измените вспомогательный алгоритм "фрагмент" (А19) так, чтобы при выполнении алгоритма (А18) Робот переместился из А в Б и закрасил клетки, отмеченные точками (рис. 23, 24):

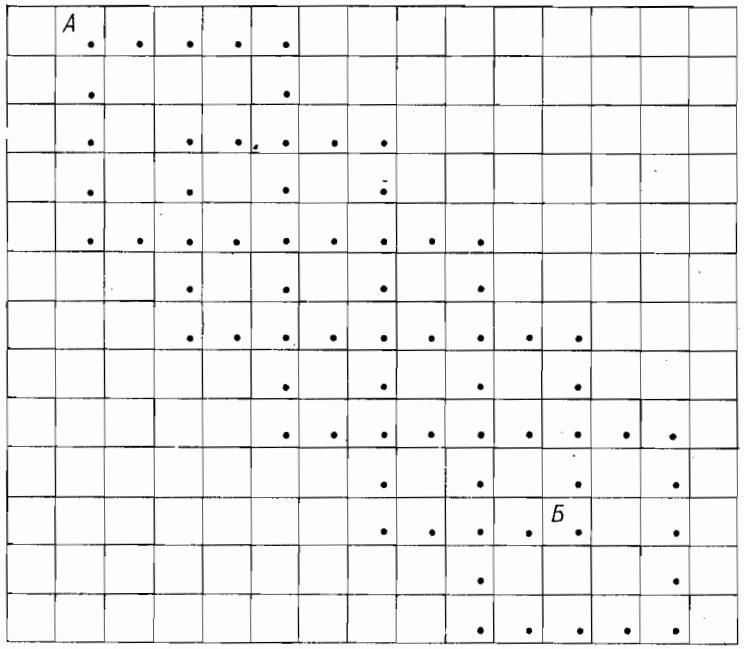
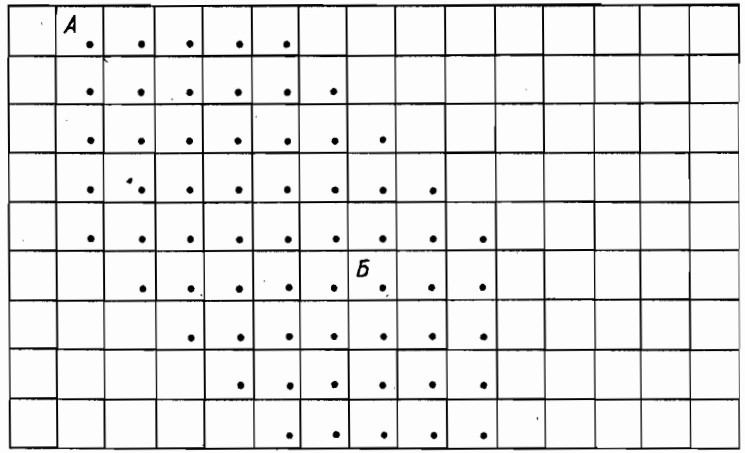
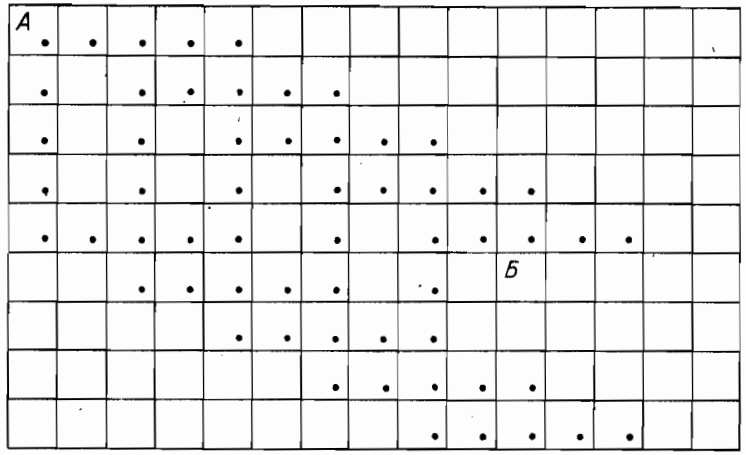


Рис. 23



а)



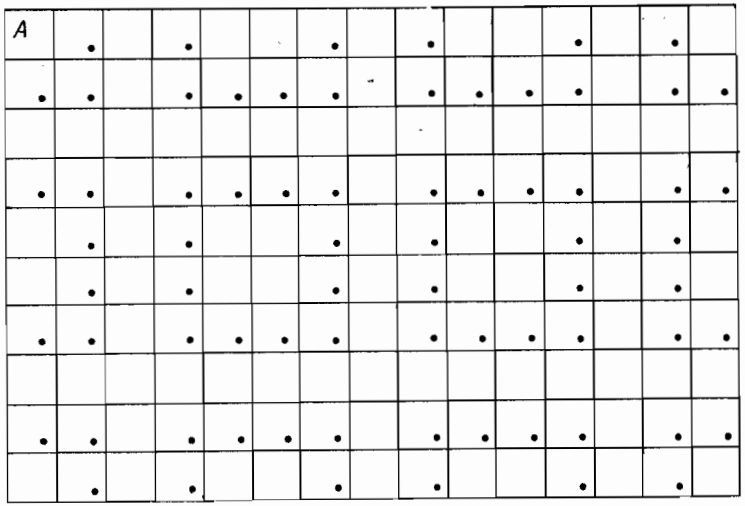
б)

Рис. 24

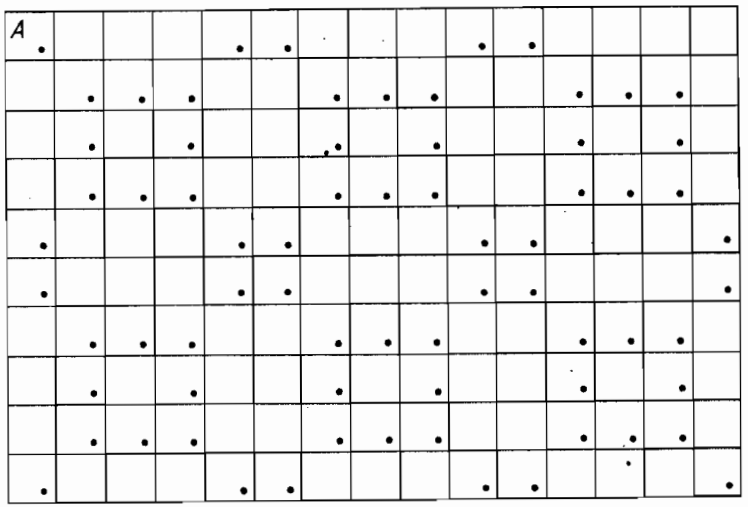
9. Известно, что вспомогательный алгоритм "фрагмент" закрашивает некоторые клетки в квадрате 4×4, причем путь Робота при выполнении этого алгоритма начинается и заканчивается в левом верхнем углу квадрата. Составьте алгоритм, который закрашивает 25 одинаковых фрагментов в квадрате 20×20 клеток.

10. Составьте алгоритм, который закрашивает в шахматном порядке квадрат 10×10, начиная с левого верхнего угла.

11. Составьте алгоритм, при выполнении которого Робот перемещается из клетки А в клетку Б и закрашивает клетки, отмеченные точками (рис. 25).



а)



б)

Рис. 25

12. Сколько команд "закрасить" ЭВМ даст Роботу при выполнении алгоритма упражнения 7? Сколько клеток при этом будет закрашено а) дважды; б) трижды?

13. Какую последовательность команд выполнит Чертежник и что будет нарисовано при вызовах: а) квадрат (0); б) квадрат (—1)?

14. Составьте алгоритм "прямоугольник (**арг вещ** x, y, a, b)", который рисует прямоугольник с длинами сторон a и b, начиная и кончая в углу — точке (x, y) (рис. 26).

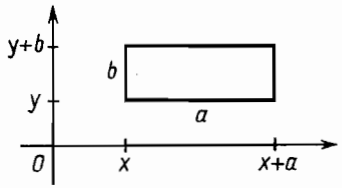
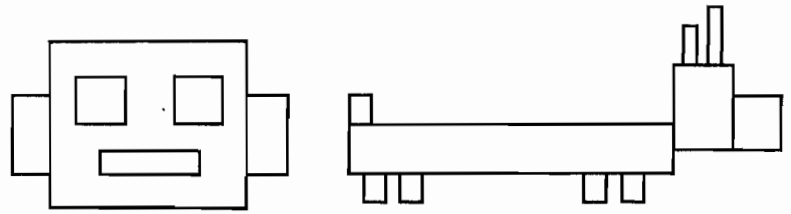


Рис. 26

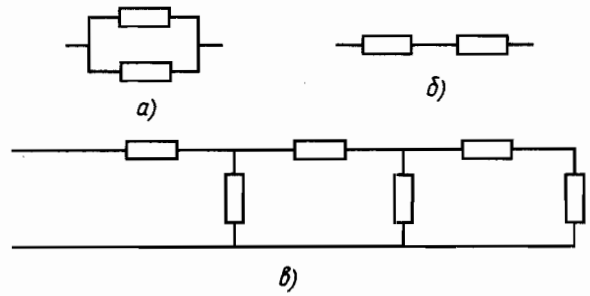
15. Используя алгоритм "прямоугольник" (упр. 14), составьте алгоритмы рисования следующих картинок (рис. 27).



а) Робот б) Собачка

Рис. 27

16. Составьте алгоритмы рисования схем (рис. 28):



а) б) в)

Рис. 28

17. Придумайте какую-нибудь картинку, составленную из прямоугольников. Составьте алгоритм для рисования этой картинки.

18 Нарисуйте результат выполнения алгоритма:

а) **алг** тоннель (А20)
нач
| квадрат (10); сместиться на вектор (1, 1)
| квадрат (7); сместиться на вектор (1, 1)
| квадрат (4); сместиться на вектор (1, 1)
| квадрат (1)
кон

б) **алг** спираль (А21)
нач
| опустить перо
| виток (1); виток (3); виток (5); виток (7); виток (9)
| поднять перо
кон

алг виток (**арг вещ** а) (А22)
нач
| сместиться на вектор (а, 0)
| сместиться на вектор (0, —а)
| сместиться на вектор (—а—1, 0)
| сместиться на вектор (0, а+1)
кон

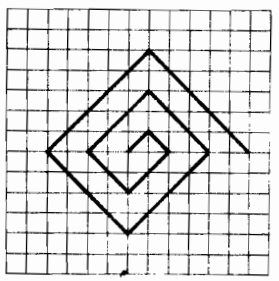


Рис. 29

19. Что нарисует Чертежник при выполнении алгоритма "спираль" (А21), если в алгоритме "виток" (А22) заменить —а—1 на —а—а и а+1 заменить на а+а?

20. Измените алгоритм "виток" (А22) так, чтобы спираль в алгоритме (А21) раскручивалась против часовой стрелки.

21. Составьте алгоритм рисования спирали, изображенной на рисунке 29.

22. Измените ваше решение упражнения 21 так, чтобы расстояние между витками при каждом новом витке увеличивалось.

23. Нарисуйте результат выполнения алгоритма "орнамент":

алг орнамент (А23)
дано | перо Чертежника в левом верхнем углу будущего орнамента размером 12×12 и поднято
надо | нарисован орнамент, перо в левом нижнем углу и поднято
нач
| ряд; ряд; ряд
кон

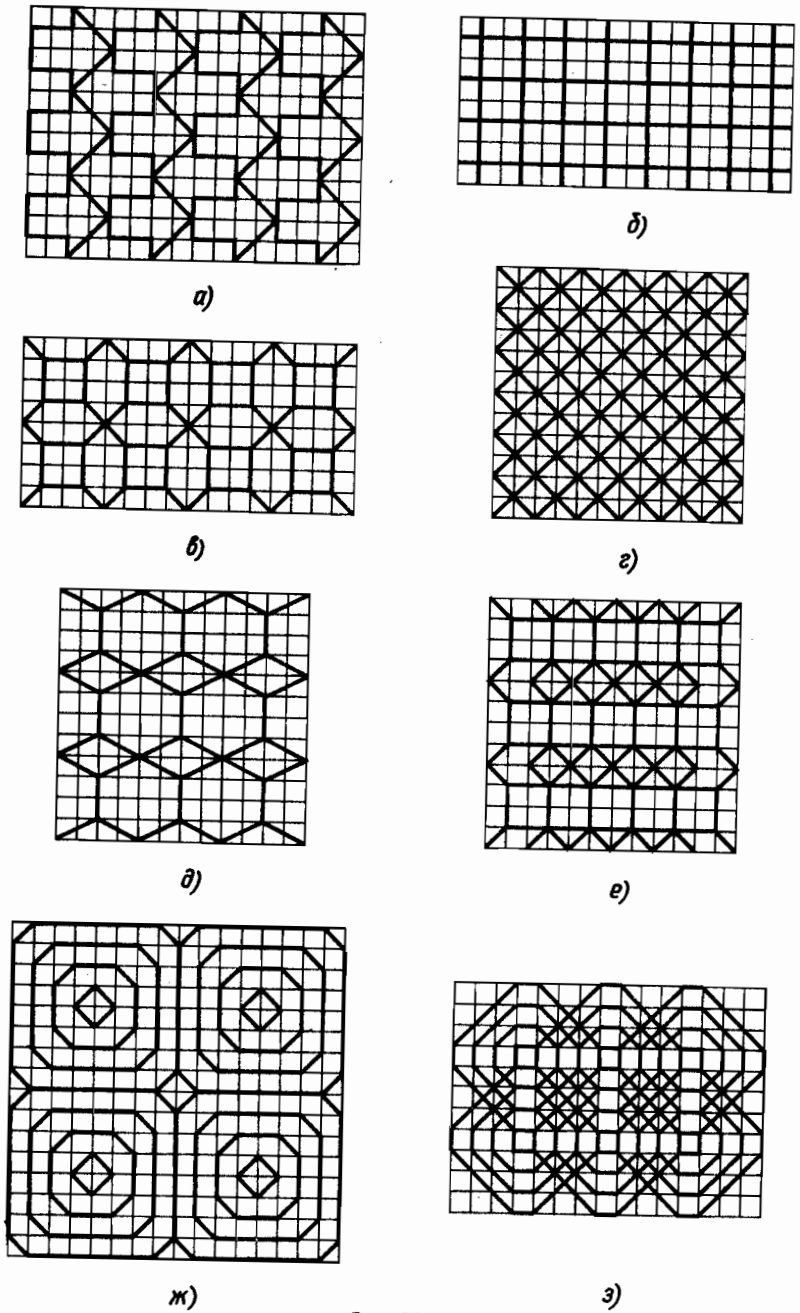


Рис. 30

алг ряд (А24)
дано | перо Чертежника в левом верхнем углу будущего ряда
| размером 12×4 и поднято
надо | нарисован ряд, перо в левом нижнем углу ряда и поднято
нач
| фрагмент; фрагмент; фрагмент
| сместиться на вектор (—12, —4)
кон

алг фрагмент (А25)
дано | перо Чертежника в левом верхнем углу будущего фрагмента размером 4×4 и поднято
надо | нарисован фрагмент, перо в правом верхнем углу и поднято
нач
| опустить перо
| сместиться на вектор (2, —2)
| сместиться на вектор (—2, —2)
| поднять перо; сместиться на вектор (4,0); опустить перо
| сместиться на вектор (0, 1)
| сместиться на вектор (—2, 0)
| сместиться на вектор (0, 2)
| сместиться на вектор (2, 0)
| сместиться на вектор (0, 1)
| поднять перо
кон

24. По образцу упражнения 23 составьте алгоритмы рисования орнаментов (рис. 30).

§ 7. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ И ПРАВИЛА ИХ ЗАПИСИ

7.1. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

В алгоритмическом языке можно использовать не только числа, но и числовые и алгебраические выражения (формулы). В информатике эти выражения называются ***арифметическими***. Правила алгоритмического языка позволяют при записи алгоритма всюду, где можно написать число, написать и произвольное арифметическое выражение. Рассмотрим пример:

алг нарисовать М размером (**арг вещ** a, b) (А26)
дано | перо Чертежника в точке А (рис. 31) и поднято
надо | нарисована буква М ширины a и высоты b,
| перо поднято и находится в точке Б (рис. 31)

нач
| опустить перо
| сместиться на вектор (0, b)
| сместиться на вектор (a/2, —b/2)
| сместиться на вектор (a/2, b/2)
| сместиться на вектор (0, —b)
| поднять перо
| сместиться на вектор (a/2, 0)
кон

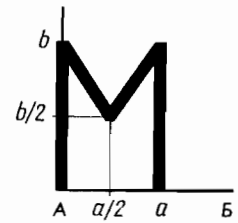


Рис. 31

Знак "/" здесь означает деление. Таким образом, запись "a/2" означает $\frac{a}{2}$, "(a+b)/c" означает $\frac{a+b}{c}$ и т. д.

7.2. ВЫРАЖЕНИЯ ВЫЧИСЛЯЕТ ЭВМ

При выполнении вызова "нарисовать М размером (3, 5)" ЭВМ запомнит, что a=3, b=5; вычислит все выражения и скомандует Чертежнику:

опустить перо
сместиться на вектор (0, 5)
сместиться на вектор (1.5, —2.5)
сместиться на вектор (1.5, 2.5)
сместиться на вектор (0, —5)
поднять перо
сместиться на вектор (1.5, 0)

Чертежник никаких вычислений не производит — он лишь выполняет последовательно поступающие команды с конкретными числовыми аргументами.

7.3. ПРАВИЛА ЗАПИСИ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Арифметические выражения в алгоритмическом языке должны быть записаны в так называемой ***линейной*** записи согласно следующим правилам:

— выражение должно быть записано в виде линейной цепочки символов (вместо x_1 и v_0 надо писать x1, v0);

— для обозначения операции умножения используется звездочка (*), для операции деления — косая черта (/), для операции возведения в степень — две звездочки (**);

— нельзя опускать знаки операций, например писать 4a. Для записи произведения чисел 4 и a надо писать 4*a;

— аргументы функций (sin, cos и др.), как и аргументы вспомогательных алгоритмов, записываются в круглых скобках, например sin (x), cos (4*a).

Линейная запись позволяет вводить выражения в память ЭВМ, последовательно нажимая на соответствующие клавиши на клавиатуре.

7.4. ТАБЛИЦА ЗНАКОВ ОПЕРАЦИЙ И СТАНДАРТНЫХ ФУНКЦИЙ АЛГОРИТМИЧЕСКОГО ЯЗЫКА

Название операции или функции		Форма записи
сложение		x+y
вычитание		x−y
умножение		x*y
деление		x/y
возведение в степень		x**y
корень квадратный	$\sqrt{x}$	sqrt (x)
абсолютная величина	$\|x\|$	abs (x)
знак числа (−1, 0 или 1)		sign (x)
синус	sin x	sin (x)
косинус	cos x	cos (x)
тангенс	tg x	tg (x)
котангенс	ctg x	ctg (x)
арксинус	arcsin x	arcsin (x)
арккосинус	arccos x	arccos (x)
арктангенс	arctg x	arctg (x)
арккотангенс	arcctg x	arcctg (x)
натуральный логарифм	ln x	ln (x)
десятичный логарифм	lg x	lg (x)
степень числа e (e≈2.718281)	e^x	exp (x)
минимум из чисел x и y		min (x, y)
максимум из чисел x и y		max (x, y)
остаток от деления x на y (x, y — целые)		mod (x, y)
частное от деления x на y (x, y — целые)		div (x, y)
целая часть x, т. е. максимальное целое число, не превосходящее x		int (x)
случайное число от 0 до x (см. § 25)		rnd (x)

7.5. ПРИМЕРЫ ЗАПИСИ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Выражение	Запись на алгоритмическом языке
$\frac{-1}{x^2}$	−1/x**2
$\frac{a}{bc}$	a/(b*c)
$\frac{a}{b}c$	a/b*c или (a/b)*c
$2^{2^{2^n}}$	2**2**2**n или 2**(2**(2**n))
x^{y^z}	x**y**z или x** (y**z)
$\frac{-b+\sqrt{b^2-4ac}}{2a}$	(−b+sqrt (b**2−4*a*c))/(2*a)
$\sqrt{p(p-a)(p-b)(p-c)}$	sqrt (p* (p−a)*(p−b)*(p−c))
$\frac{a+b+c}{2}$	(a+b+c)/2
$\sqrt{a^2+b^2-2ab\cos\gamma}$	sqrt (a**2+b**2−2*a*b*cos (γ))
$\frac{ad+bc}{bd}$	(a*d+b*c)/(b*d)
$\frac{\operatorname{tg}\alpha}{\sqrt{1+\operatorname{tg}^2\alpha}}$	tg (α)/sqrt (1+tg (α)**2)
$\sin\alpha\cos\beta+\cos\alpha\sin\beta$	sin (α)*cos (β)+cos (α)*sin (β)

УПРАЖНЕНИЯ

1. Переведите из линейной записи в обычную:

а) a/b/c; г) a/b**c; ж) a/b**c**d;
б) a*b/c; д) a+b/c; з) 1/(1+x*x);
в) a/b*c; е) (a+b)/c; и) 1/(1+x**2).

2. Переведите из линейной записи в обычную:

а) 1/sqrt (1+x**2); е) sin (x)**2+sin (y)**2;
б) sqrt (x**2+y**2); ж) sin (x**2)+sin (y**2);
в) x**(1/3); з) a+b/c+d;
г) x**(−1/3); и) (a+b)/(c+d);
д) 1/x**(1/3); к) a/sin (A).

3. Переведите из линейной записи в обычную:

а) sqrt (tg (A+B))/sqrt (tg (A−B));
б) 1/2*a*b*sin (C);
в) sqrt (b**2+c**2+2*b*c*cos (A))/2;
г) 2*b*c*cos (A/2)/(b+c);
д) sqrt ((p−a)*(p−b)*(p−c)*p);
е) 4*R*sin (A/2)*sin (B/2)*sin (C/2);
ж) (a*x+b)/(c*x+d);
з) sqrt (a*x**2+b*x+c);
и) arctg (x/sqrt (1−x**2));
к) 2*sin ((α+β)/2)*cos ((α−β)/2).

4. Запишите по правилам алгоритмического языка следующие выражения:

а) $\sqrt{x_1^2+x_2^2}$;
б) $x_1x_2+x_1x_3+x_2x_3$;
в) $v_0t+\frac{at^2}{2}$;
г) $\frac{mv^2}{2}+mgh$;
д) $\frac{1}{R_1}+\frac{1}{R_2}$;
е) $mg\cos\alpha$;
ж) $2\pi R$;
з) b^2-4ac;
и) $\gamma\frac{m_1m_2}{r^2}$;
к) $|x|+|x+1|$;
л) I^2R;
м) $ab\sin C$;
н) $\sqrt{a^2+b^2-2ab\cos C}$;
о) $\sin x\cos y+\sin y\cos x$;
п) $\frac{ad+bc}{bd}$;
р) $a\sin\alpha+b\cos\alpha$;
с) $\sqrt{1-\sin^2x}$;
т) $\frac{1}{\sqrt{ax^2+bx+c}}$;
у) $\frac{\sqrt{x+1}-\sqrt{x-1}}{2\sqrt{x}}$;
ф) $|1-|x||$.

§ 8. КОМАНДЫ АЛГОРИТМИЧЕСКОГО ЯЗЫКА. ЦИКЛ N РАЗ

8.1. ЦИКЛ N РАЗ

При составлении алгоритмов довольно часто встречаются случаи, когда некоторую последовательность команд нужно выполнить несколько раз подряд (см., например, алгоритм А2). Для упрощения записи алгоритма в таких случаях можно использовать специальную ***команду алгоритмического языка*** — цикл n **раз**:

```
алг из А и Б                                          (А27)
 дано |Робот в клетке А (рис. 6)
 надо |Робот в клетке Б (рис. 6)
нач
| нц 5 раз
|  | вверх; вверх; вправо; вниз; вниз; вправо
| кц
кон
```

При выполнении этого алгоритма ЭВМ 5 раз повторит последовательность вызовов «вверх; вверх; вправо; вниз; вниз; вправо» и Робот окажется в клетке Б (рис. 6).

Команда n **раз** называется ***циклом***, поскольку при ее выполнении циклически повторяется одна и та же последовательность команд.

8.2. ОБЩИЙ ВИД ЦИКЛА N РАЗ

В общем виде цикл n **раз** записывается так:

```
нц число повторений раз
| тело цикла (последовательность команд)
кц
```

Служебные слова **нц** (начало цикла) и **кц** (конец цикла) пишутся строго одно под другим и соединяются вертикальной чертой. Правее этой черты записывается повторяемая последовательность команд ***(тело цикла)***. При выполнении алгоритма эта последовательность команд циклически повторяется указанное число раз. Правила алгоритмического языка допускают задание любого целого числа повторений. Оно может быть нулевым и даже отрицательным. Эти случаи не считаются ошибочными — просто тело цикла не будет выполнено ни разу, а ЭВМ сразу перейдет к выполнению команд, записанных после **кц**.

8.3. ПРОСТЫЕ И СОСТАВНЫЕ КОМАНДЫ

Рассмотренные нами раньше команды записывались на одной строке. Команда n **раз** выглядит по-другому. Она занимает несколько строк и, что самое главное, содержит в себе другие команды алгоритмического языка. Команды подобного рода называются ***составными***, в отличие от изученной нами ранее ***простой*** команды вызова.

Кроме вызова и цикла n **раз**, в алгоритмическом языке есть и другие команды: цикл **пока**, команды ветвления **если** и **выбор**, команда присваивания, команды ввода/вывода и др.

8.4. ПРИМЕР ИСПОЛЬЗОВАНИЯ ЦИКЛА N РАЗ

Используя цикл n **раз**, можно составить алгоритмы для смещения Робота на заданное число клеток в нужном направлении, например:

```
алг вверх на (арг цел n)                              (А28)
 дано |на поле Робота стен нет
 надо |Робот сместился на n клеток вверх
нач
| нц n раз
|  | вверх
| кц
кон
```

Аналогично можно составить алгоритмы "вниз на (**арг цел** n)", "вправо на (**арг цел** n)" и "влево на (**арг цел** n)". В дальнейшем

мы часто будем пользоваться этими алгоритмами как вспомогательными.

Обратите внимание, что, не используя цикл n **раз**, алгоритм (А28) составить нельзя, т. е. цикл n **раз** не только сокращает запись, но и позволяет решать новые, более сложные задачи.

8.5. ЧТО ЗНАЧИТ ПОВТОРИТЬ КОМАНДУ «— 10 РАЗ»!

Если в основном алгоритме написать вызов "вверх на (—10)", то при выполнении алгоритма А28 ЭВМ запомнит, что n = —10, и при выполнении цикла

нц n **раз**
| вверх
кц

в соответствии с правилами алгоритмического языка ***не выполнит тело цикла ни разу*** (т. е. Робот вообще не получит никаких команд и не сдвинется с места).

8.6. СЕРИЯ КОМАНД В ЦИКЛЕ МОЖЕТ СОСТОЯТЬ ИЗ НЕСКОЛЬКИХ КОМАНД

алг закрасить ряд (**арг цел** m) (А29)
дано |на поле Робота стен нет, Робот в клетке А
надо |Робот закрасил m клеток вправо от исходного положения
|и вернулся в клетку А (рис. 32)

нач
| **нц** m **раз**
| | закрасить; вправо
| **кц**
| влево на (m)
кон

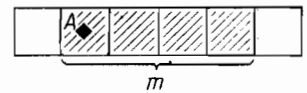


Рис. 32

8.7. КОРОТКИЕ АЛГОРИТМЫ МОГУТ ОПИСЫВАТЬ ДЛИННЫЕ ПОСЛЕДОВАТЕЛЬНОСТИ ДЕЙСТВИЙ

Если написать вызов "закрасить ряд (4)", то при его выполнении ЭВМ подставит m = 4 и выдаст Роботу 12 команд:

закрасить; вправо
закрасить; вправо
закрасить; вправо
закрасить; вправо
влево; влево; влево; влево

При вызове "закрасить ряд (1000)" ЭВМ запомнит, что m = 1000, и выдаст Роботу 3000 команд. А ведь в записи алгоритма "закрасить ряд" всего 10 строк!

Таким образом, короткие алгоритмы могут описывать очень длинные последовательности действий.

8.8. ВНУТРИ ЦИКЛА МОЖНО ВЫЗЫВАТЬ ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ

алг закрасить прямоугольник (**арг цел** m, n) (А30)
дано |на поле Робота стен
|нет, Робот в клетке А
надо |закрашен прямоугольник
|размером m*n (рис. 33),
|Робот в клетке А

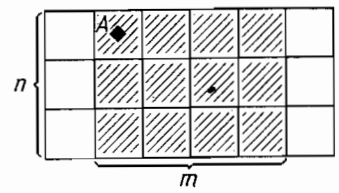


Рис. 33

нач
| **нц** n **раз**
| | закрасить ряд (m)
| | вниз
| **кц**
| вверх на (n)
кон

Внутри цикла в этом алгоритме n раз вызывается вспомогательный алгоритм "закрасить ряд" (А29).

УПРАЖНЕНИЯ

1. Сколько клеток будет закрашено и сколько команд ЭВМ выдаст Роботу при выполнении вызова
а) закрасить прямоугольник (1, 1);
б) закрасить прямоугольник (0, 11);
в) закрасить прямоугольник (9, 0);
г) закрасить прямоугольник (9, 11)?

2. Опишите, как будет выполняться вызов "закрасить прямоугольник (3, 3)" в ситуациях, изображенных на рисунке 34.

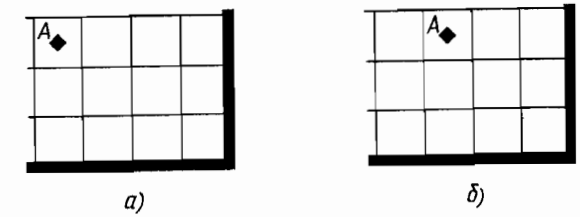


Рис. 34

3. Измените алгоритмы "закрасить ряд" (А29) и "закрасить прямоугольник" (А30) так, чтобы при вызове "закрасить прямоугольник (3, 3)" в ситуации на рисунке 34, *б* отказа не возникало, а оказался закрашенным квадрат 3 на 3 клетки.

4. Составьте алгоритм "горизонтальная ломаная (**арг цел** n, **арг вещ** a)", рисующий с помощью Чертежника ломаную линию с 2n звеньями, показанную на рисунке 35.

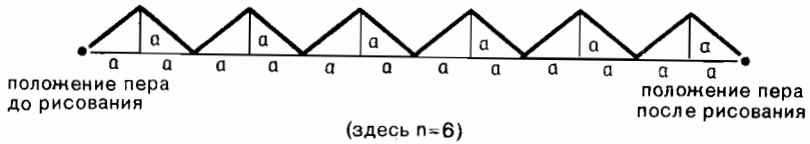


Рис. 35

5. Составьте алгоритм "вертикальная ломаная (**арг цел** n, **арг вещ** a)", рисующий ломаную упражнения 4, повернутую вокруг начального положения пера на 90° по часовой стрелке.

6. Используя алгоритмы из упражнений 4 и 5 как вспомогательные, составьте алгоритмы, рисующие: а) m горизонтальных ломаных с 2n звеньями одна под другой на расстоянии *b* друг от друга; б) m вертикальных ломаных с 2n звеньями одна под другой на расстоянии b друг от друга.

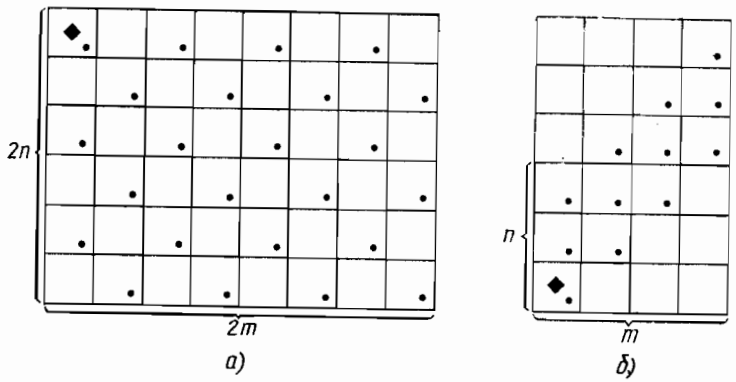
7. ЭВМ выполнила последовательность команд:

горизонтальная ломаная (5, 1)
вертикальная ломаная (7, 1)
горизонтальная ломаная (5, −1)
вертикальная ломаная (7, −1)

Что нарисовал Чертежник?

8. Вспомогательный алгоритм "картинка" рисует некоторую картинку в квадрате 1×1 и возвращает перо в начальное положение — левый нижний угол квадрата. Составьте алгоритм, рисующий 100 экземпляров картинки в квадрате 10×10.

9. Составьте алгоритм с целыми аргументами m и n, который с помощью Робота закрашивает следующие клетки (рис. 36):



а) б)

Рис. 36

§ 9. АЛГОРИТМЫ С «ОБРАТНОЙ СВЯЗЬЮ». КОМАНДА ПОКА

9.1. КОМАНДЫ «ОБРАТНОЙ СВЯЗИ»

До сих пор все составленные нами алгоритмы управления Роботом приводили к выполнению определенной, заранее известной последовательности действий. ЭВМ лишь приказывала Роботу выполнить какие-то действия, но никак не анализировала их результаты и обстановку на поле Робота.

Представьте себе начальника, который отдает приказы, но не получает никаких донесений о результатах их выполнения; повара, который не может попробовать приготавливаемое блюдо; шофера, ведущего автомобиль с закрытыми глазами. Понятно, что так далеко не уедешь; если мы хотим составлять алгоритмы для решения сложных задач, то надо уметь не только командовать Роботом, но и анализировать обстановку, в которой он оказался. Для этого у Робота есть 12 команд "обратной связи", при вызове которых Робот сообщает информацию об обстановке вокруг себя:

лог сверху стена	**лог** сверху свободно
лог снизу стена	**лог** снизу свободно
лог справа стена	**лог** справа свободно
лог слева стена	**лог** слева свободно
лог клетка закрашена	**лог** клетка не закрашена

вещ температура
вещ радиация

В ответ на команды, отмеченные служебным словом **вещ** (вещественный), Робот сообщает число: температуру или уровень радиации в той клетке, где он стоит. В ответ на команды, отмеченные служебным словом **лог** (логический), Робот отвечает **да** или **нет**. Например, в обстановке на рисунке 4 на вопрос "слева стена" Робот ответит **нет**, на вопрос "сверху стена" ответит **да**, на вопрос "клетка закрашена" ответит **нет**.

9.2. ИСПОЛЬЗОВАНИЕ КОМАНД «ОБРАТНОЙ СВЯЗИ» ПРИ УПРАВЛЕНИИ РОБОТОМ «ВРУЧНУЮ»

Пусть где-то ниже Робота на неизвестном расстоянии есть стена. Нужно подвести Робота вплотную к этой стене. Как это сделать?

Если мы командуем Роботом вручную, без ЭВМ, то проблем нет: надо спросить у Робота "снизу свободно?". Если Робот ответит **нет**, значит, он уже у стены и задача решена. Если же Робот ответит **да**, то нужно скомандовать "вниз" и опять спросить "снизу

свободно?". Если Робот ответит **да** — опять командовать "вниз", спрашивать "снизу свободно?" и т. д., пока Робот не ответит **нет**. Скомандуем ли мы при этом "вниз" 0, 3, 8 или 1990 раз, заранее неизвестно — это зависит от начального расположения Робота относительно стены.

9.3. ЦИКЛ ПОКА

Наша цель, однако, не ручное управление Роботом, а составление алгоритма для ЭВМ. Поэтому приведенную выше последовательность действий надо описать на алгоритмическом языке. Алгоритм должен быть универсальным, т. е. не должен зависеть от расстояния между Роботом и стеной. Для этого в алгоритмическом языке есть специальная составная команда — цикл **пока**:

```
алг вниз до стены                                  (А31)
нач
| нц пока снизу свободно
| | вниз
| кц
кон
```

Аналогично можно составить алгоритмы "влево до стены", "вправо до стены", "вверх до стены", которыми мы далее часто будем пользоваться как вспомогательными.

9.4. ДИАЛОГ ЭВМ — РОБОТ ПРИ ВЫПОЛНЕНИИ ЦИКЛА ПОКА

Пусть в начальный момент Робот находится в клетке А (рис. 37). Тогда при выполнении алгоритма "вниз до стены" (А31) диалог ЭВМ — Робот будет таким:

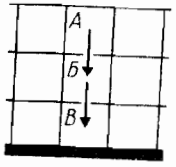


Рис. 37

ЭВМ: снизу свободно? ***Робот:*** **да**
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Б)
ЭВМ: снизу свободно? ***Робот:*** **да**
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку В)
ЭВМ: снизу свободно? ***Робот:*** **нет**

Поскольку Робот ответил **нет**, т. е. записанное после **пока** условие не соблюдается, то на этом выполнение цикла **пока** и алгоритма "вниз до стены" заканчивается.

9.5. ОБЩИЙ ВИД ЦИКЛА ПОКА

В общем виде цикл **пока** записывается так:

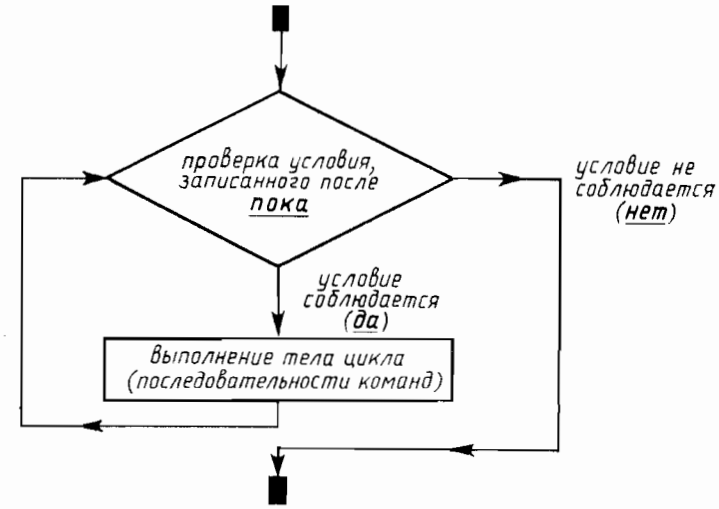
```
нц пока условие
| тело цикла (последовательность команд)
кц
```

При его выполнении ЭВМ циклически повторяет следующие действия:

а) проверяет записанное после служебного слова **пока** условие;

б) если условие не соблюдается (Робот ответил **нет**), то выполнение цикла завершается и ЭВМ начинает выполнять команды, записанные после **кц**. Если же условие соблюдается (Робот ответил **да**), то ЭВМ выполняет тело цикла, снова проверяет условие и т. д.

9.6. ГРАФИЧЕСКАЯ СХЕМА ВЫПОЛНЕНИЯ ЦИКЛА ПОКА



9.7. ТЕЛО ЦИКЛА МОЖЕТ НЕ ВЫПОЛНИТЬСЯ НИ РАЗУ

Если условие в цикле **пока** не соблюдается с самого начала, то тело цикла не выполняется ни разу. Например, если в алгоритме "вниз до стены" (А31) Робот на первый же вопрос "снизу свободно" ответит **нет** (т. е. если снизу от него с самого начала будет стена), то ЭВМ не вызовет команду "вниз" ни разу.

9.8. ЗАЦИКЛИВАНИЕ

Выполнение цикла **пока** может и не завершиться, если условие все время будет соблюдаться (эту ситуацию принято называть ***зацикливанием***). Например, если ниже Робота никаких стен вообще нет, то при выполнении алгоритма "вниз до стены" (А31) ЭВМ "зациклится", т. е. будет бесконечно спрашивать у Робота "снизу свободно", получать в ответ **да** и командовать "вниз".

9.9. УСЛОВИЕ ЦИКЛА НЕ ПРОВЕРЯЕТСЯ В ПРОЦЕССЕ ВЫПОЛНЕНИЯ ТЕЛА ЦИКЛА

Условие в цикле **пока** проверяется только ***перед*** выполнением тела цикла, но не проверяется в процессе выполнения. Приведем два примера.

Пример 1. Пусть Робот находится в клетке А (рис. 38) и ЭВМ выполняет цикл

нц пока снизу свободно
| вниз
| вниз
кц

Тогда диалог ЭВМ — Робот будет таким:

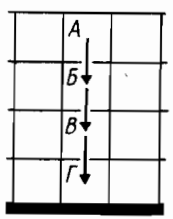


Рис. 38

ЭВМ: снизу свободно? ***Робот:*** **да**
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Б)
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку В)
ЭВМ: снизу свободно? ***Робот:*** **да**
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Г)
ЭВМ: вниз ***Робот:*** отказ

Таким образом, в процессе выполнения тела цикла ЭВМ условие ***не проверяет*** и вопросов Роботу не задает. Если в процессе выполнения тела цикла условие перестанет соблюдаться, то ЭВМ про это не узнает и будет пытаться выполнить тело цикла до конца.

Пример 2. Пусть Робот находится в клетке А (рис. 39) и ЭВМ выполняет цикл

нц пока клетка закрашена
| вниз
| вниз
| вниз
кц

Тогда диалог ЭВМ — Робот будет таким:

ЭВМ: клетка закрашена? ***Робот:*** **да**
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Б)
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку В)
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Г)
ЭВМ: клетка закрашена? ***Робот:*** **да**
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Д)
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Е)
ЭВМ: вниз ***Робот:*** смещается вниз (в клетку Ж)
ЭВМ: клетка закрашена? ***Робот:*** **нет**

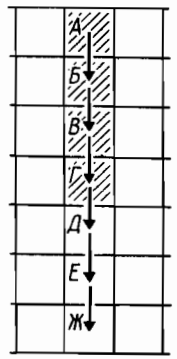


Рис. 39

Таким образом, в процессе второго выполнения тела цикла условие "клетка закрашена" перестанет соблюдаться (клетки Д и Е не закрашены). Однако ЭВМ доведет выполнение тела цикла до конца и Робот окажется на две клетки ниже клетки Д.

9.10. ЗАКРАШИВАНИЕ РЯДА

Дано, что Робот находится у левой стены внутри прямоугольника, огороженного со всех сторон стенами (рис. 40). Внутри прямоугольника стен нет, размеры прямоугольника неизвестны. Требуется закрасить горизонтальный ряд клеток от исходного положения Робота до правой стены и вернуть Робота в исходное положение.

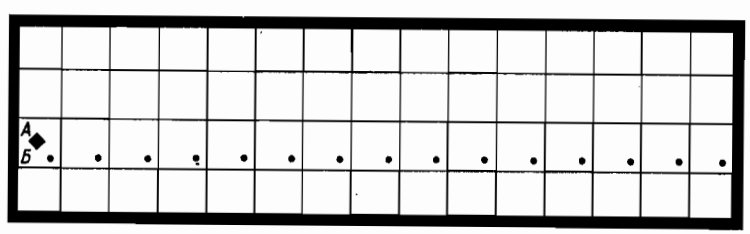


Рис. 40

Одно из возможных решений таково: сначала будем двигать Робота вправо до стены, закрашивая по дороге клетку за клеткой, а потом вернем его обратно (влево до стены). Попробуем записать первую часть:

```
нц пока справа свободно
| вправо
| закрасить
кц
```

При выполнении этого цикла окажутся закрашенными все клетки правее исходного положения Робота, но сама эта клетка останется незакрашенной. Поэтому перед выполнением цикла нужно отдельно закрасить исходную клетку:

```
алг закрасить ряд вправо и вернуться                    (А32)
 дано |Робот стоит у левой стены внутри огороженного с четырех
      |сторон прямоугольника (рис. 40)
 надо |горизонтальный ряд, в левой клетке которого стоял Робот,
      |полностью закрашен. Робот в исходном положении
нач
| закрасить
| нц пока справа свободно
| | вправо
| | закрасить
| кц
| влево до стены |возвращение в исходное положение
кон
```

9.11. СОСТАВЛЕНИЕ АЛГОРИТМОВ С ЦИКЛОМ ПОКА

Всякий раз, когда число повторений каких-то действий заранее неизвестно, используется цикл **пока**. Составление таких циклов — трудная задача. Короткий цикл **пока** может описывать длинную последовательность действий. Чтобы не запутаться и что-нибудь не забыть, лучше всего придумывать цикл **пока** по частям:

1) Понять, когда цикл должен закончиться, т. е. сформулировать ***условие окончания*** цикла. Записать после **пока** противоположное условие — ***условие продолжения*** цикла.

2) Выяснить, что и как будет меняться в цикле, описать промежуточные состояния после нескольких повторений цикла.

3) Описать, что происходит при однократном выполнении цикла (принято говорить "за один ***шаг*** цикла"), т. е. записать тело цикла.

4) Проверить, что цикл рано или поздно закончится, а не будет повторяться вечно.

9.12. ЗАКРАШИВАНИЕ КОРИДОРА ПРОИЗВОЛЬНОЙ ДЛИНЫ

Дано, что Робот стоит в левом конце горизонтального коридора, нижняя стена которого сплошная, а в верхней имеется несколько выходов. Надо составить алгоритм, который переводит

Робота из клетки А в клетку Б (рис. 41) и закрашивает все клетки коридора (на рисунке 41 отмечены точками).

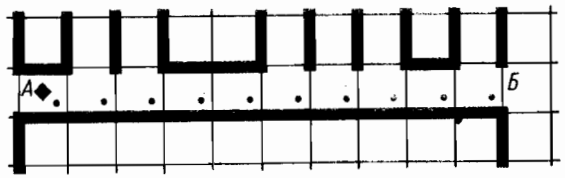


Рис. 41

Как составить такой алгоритм? Поскольку число клеток в коридоре неизвестно, то при записи алгоритма не обойтись без цикла **пока**. В какой же момент цикл должен окончиться? Он должен окончиться, когда Робот окажется в клетке Б. Чем же клетка Б отличается от клеток коридора? Как видно из рисунка 41, у клетки Б нет стены снизу, а у всех клеток коридора стена снизу есть. Поэтому после **пока** можно написать условие "снизу стена". За один шаг цикла Робот должен закрашивать очередную клетку и переходить в следующую:

```
алг закрасить коридор                                   (А33)
 дано |Робот в левой клетке горизонтального коридора (рис. 41)
 надо |Робот вышел из коридора вправо, коридор закрашен
нач
| нц пока снизу стена
| | закрасить
| | вправо
| кц
кон
```

9.13. ВХОД В РАДИОАКТИВНУЮ ЗОНУ

```
алг вход в радиоактивную зону (арг вещ а)               (А34)
 дано |на поле Робота стен нет, ниже Робота расположена
      |радиоактивная зона
 надо |Робот сместился вниз в первую клетку, уровень радиации
      |в которой превышает а
нач
| нц пока радиация ⩽ а
| | вниз
| кц
кон
```

Здесь после **пока** записано условие "радиация $\leqslant$ а". При проверке этого условия ЭВМ вызывает команду Робота "радиация" и полученный ответ (уровень радиации в той клетке, где находится Робот) сравнивает со значением аргумента а.

9.14. ВЫХОД В ЛЕВЫЙ ВЕРХНИЙ УГОЛ В ЛАБИРИНТЕ

Робот находится внутри прямоугольного лабиринта, огороженного с четырех сторон стенами. Внутри лабиринта стены имеют вид отрезков прямых, не касаются друг друга и наружных стен (рис. 42). Надо составить алгоритм, который в любом таком лабиринте перемещает Робота в левый верхний угол.

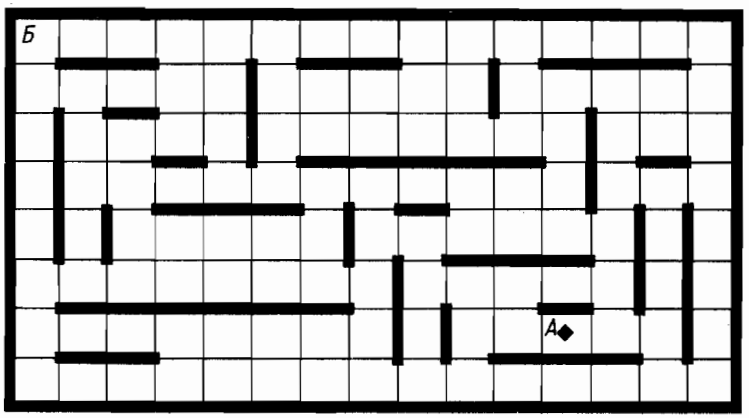


Рис. 42

Поскольку размеры лабиринта неизвестны, то нам не обойтись без цикла **пока**. Когда этот цикл должен закончиться, т. е. чем конечное положение Робота (клетка Б) отличается от всех остальных? Видно, что в клетке Б стены есть и слева и сверху, а во всех остальных клетках хотя бы одной из этих стен нет. Значит, условие окончания цикла — "сверху стена **и** слева стена", т. е. цикл надо продолжать до тех пор, пока либо слева, либо сверху от клетки свободно. Это условие записывается так: "слева свободно **или** сверху свободно" (общий вид таких составных условий будет рассмотрен в § 10).

Внутри цикла надо смещать Робота по направлению к углу. Воспользуемся методом последовательного уточнения — введем вспомогательный алгоритм "сместиться к углу" (его мы составим потом) и запишем основной алгоритм:

```
алг в левый верхний угол лабиринта                    (А35)
 дано |Робот где-то в лабиринте без углов (рис. 42)
 надо |Робот в левом верхнем углу лабиринта
нач
| нц пока слева свободно или сверху свободно
| | сместиться к углу
| кц
кон
```


Теперь составим вспомогательный алгоритм "сместиться к углу":

```
алг сместиться к углу                                 (А36)
 дано |Робот где-то в лабиринте без углов (рис. 42),
      |либо слева, либо сверху от Робота свободно
 надо |Робот сместился к левому верхнему углу
нач
| вверх до стены
| влево до стены
кон
```

Как же проверить, что цикл в алгоритме А35 рано или поздно закончится? Можно рассуждать так: при каждом выполнении тела цикла (т. е. при каждом выполнении алгоритма "сместиться к углу") расстояние от Робота до левого верхнего угла лабиринта уменьшается. Значит, рано или поздно Робот окажется в углу и цикл закончится.

Заметим, что Робота можно провести той же дорогой и не используя составных условий, например так:

```
алг в левый верхний угол лабиринта                    (А37)
 дано |Робот в лабиринте, вид которого изображен на рисунке 42
 надо |Робот в левом верхнем углу лабиринта
нач
| вверх до стены
| влево до стены
| нц пока сверху свободно
| | вверх до стены
| | влево до стены
| кц
кон
```

УПРАЖНЕНИЯ

1. Переделайте алгоритм "закрасить ряд вправо и вернуться" (А32), используя в нем цикл:

```
нц пока справа свободно
| закрасить; вправо
кц
```

2. Используя в качестве вспомогательного алгоритм "закрасить ряд вправо и вернуться" (А32), составьте алгоритм:

```
алг закрасить прямоугольник
 дано |Робот стоит в левом верхнем углу внутри огороженного
      |с четырех сторон прямоугольника
 надо |весь прямоугольник закрашен,
      |Робот в исходном положении
```

3. Решите упражнение 2, считая, что про начальное положение Робота в прямоугольнике ничего не известно.

4. Какие команды ЭВМ будет давать Роботу при выполнении цикла

а) **нц пока** клетка **не** закрашена
| закрасить
кц

б) **нц пока** клетка закрашена
| закрасить
кц

в ситуации, когда Робот стоит: 1) в закрашенной клетке, 2) в незакрашенной?

5. Расположение Робота показано на рисунке 43. Как будет выполняться цикл

нц пока сверху свободно
| вправо
кц

Рис. 43

6. На поле Робота стен нет. Робот находится в левом верхнем углу прямоугольника из закрашенных клеток. Составьте алгоритм, переводящий Робота в правый нижний угол прямоугольника.

7. Составьте алгоритмы со следующими заголовками:

а) **алг** закрасить до стены вправо и вернуться
дано | где-то правее Робота есть стена
надо | закрашен ряд клеток между Роботом и стеной (рис. 44),
| Робот в исходном положении

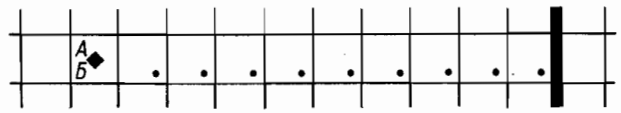


Рис. 44

б) **алг** закрасить до закрашенной клетки вправо и вернуться
дано | где-то правее Робота есть закрашенная клетка (рис. 45)
надо | закрашен ряд клеток между Роботом и этой клеткой,
| Робот в исходном положении (рис. 45)

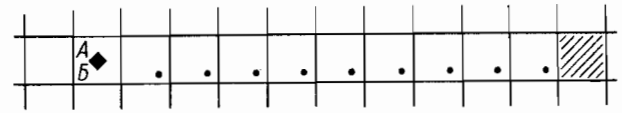


Рис. 45

в) **алг** закрасить коридор
дано | Робот где-то в горизонтальном коридоре
надо | закрашены все клетки коридора, кроме стартовой
| (клетки А), Робот в исходном положении (рис. 46)

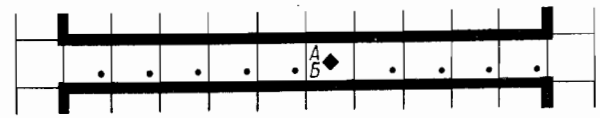


Рис. 46

г) **алг** закрасить коридор
дано | Робот где-то в горизонтальном коридоре
надо | закрашены все клетки коридора,
| Робот в исходном положении (рис. 47)

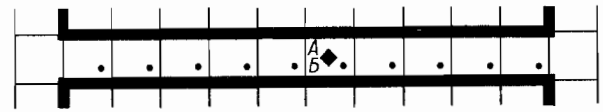


Рис. 47

д) **алг** закрасить угол
дано | Робот внутри прямоугольника, огороженного стенами
надо | закрашены все клетки правее и выше стартовой (рис. 48),
| Робот в исходном положении

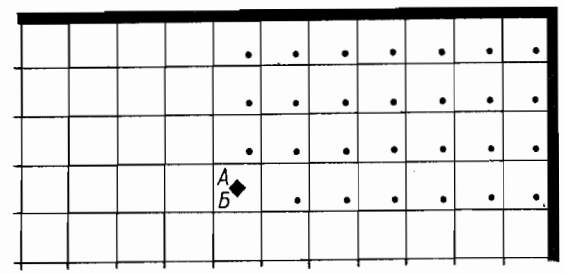


Рис. 48

8. Дано, что на поле Робота только одна стена и эта стена расположена строго горизонтально. Робот находится в одной из клеток, прилегающих к стене сверху (рис. 49). Точные размеры стены и точное расположение Робота неизвестны. Составьте алгоритм, при выполнении которого Робот:

а) окажется в одной из клеток, прилегающих к стене снизу;
б) закрасит все клетки, прилегающие к стене сверху;
в) закрасит все клетки, прилегающие к стене снизу;
г) закрасит все прилегающие к стене клетки.

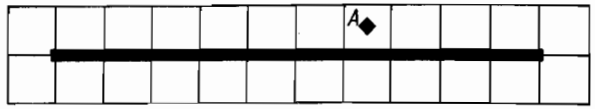


Рис. 49

9. Робот находится внутри прямоугольника, огороженного с четырех сторон стенами. Внутри прямоугольника стен нет. Составьте алгоритм, при выполнении которого Робот закрашивает все клетки внутри прямоугольника, прилегающие к стенам.

10. Составьте алгоритм со следующим заголовком:

алг обход прямоугольника
дано | Робот над верхней стороной прямоугольника, огороженного стенами; снаружи прямоугольника стен нет
надо | Робот под нижней стороной прямоугольника

11. Дано, что Робот находится в левом верхнем углу в прямоугольнике, огороженном с четырех сторон стенами. Внутри прямоугольника имеется горизонтальная стена с одним проходом, идущая от левого до правого края прямоугольника (проход не прилегает ни к левой, ни к правой стене прямоугольника). Составьте алгоритм при выполнении которого Робот переместится в правый нижний угол прямоугольника (рис. 50).

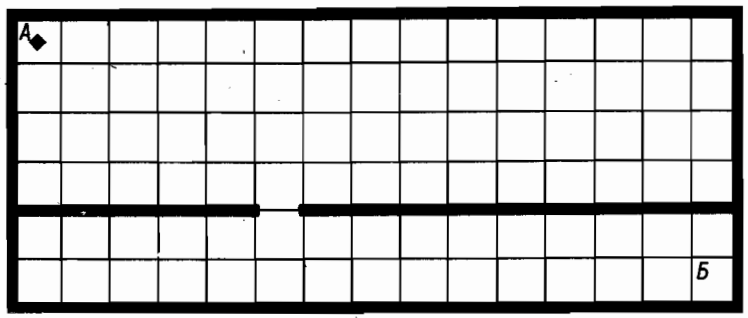


Рис. 50

12. Составьте алгоритм для закраски всех клеток вокруг прямоугольной стены (рис. 51).

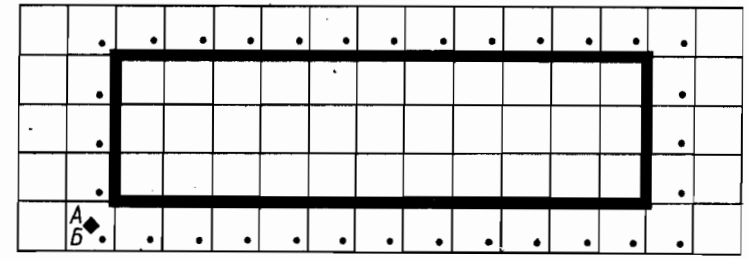


Рис. 51

13. Составьте алгоритм для закраски всех клеток вокруг Т-образной стены неизвестного размера (рис. 52).

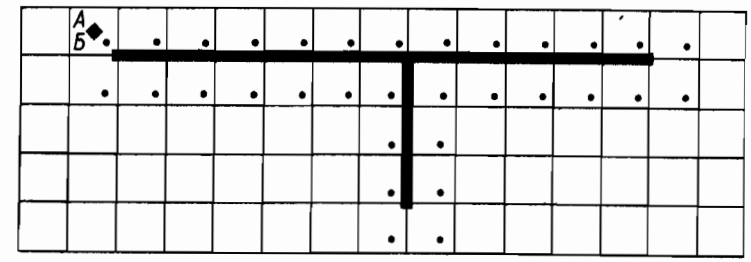


Рис. 52

14. Опишите работу алгоритма "в левый верхний угол лабиринта" (А35) в ситуации, показанной на рисунке 42, если вспомогательный алгоритм "сместиться к углу" (А36) изменен так:

а)
нач
| влево до стены
| вверх до стены
кон

б)
нач
| вверх до стены
| влево до стены
| вверх до стены
кон

§ 10. УСЛОВИЯ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ. КОМАНДЫ ЕСЛИ И ВЫБОР. КОМАНДЫ КОНТРОЛЯ

Команды "если" и "выбор" позволяют при выполнении алгоритма выбрать один из двух или нескольких вариантов действий ЭВМ.

10.1. ПРИМЕР АЛГОРИТМА С КОМАНДОЙ ЕСЛИ

Рассмотрим такую задачу: Робот стоит в левом конце горизонтального коридора, нижняя стена которого сплошная, а в верхней имеется несколько выходов. Надо составить алгоритм, который переводит Робота из клетки А в клетку Б и закрашивает все клетки коридора, из которых есть выход вверх (рис. 53). Длина коридора, количество выходов и их точное расположение заранее неизвестны.

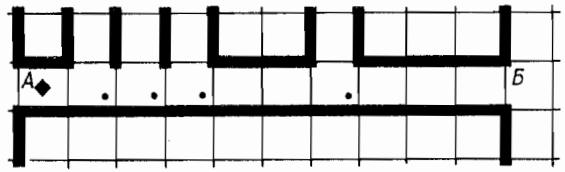


Рис. 53

Вспомним алгоритм А33 из предыдущего параграфа, решающий аналогичную задачу, но с закрашиванием всех клеток коридора. Новая задача отличается только тем, что красить надо не все клетки коридора, а лишь те, где есть выход вверх. Другими словами, если сверху свободно, то клетку надо закрасить, иначе — красить не надо. Для записи таких действий в алгоритмическом языке есть специальная составная команда **если**:

```
если сверху свободно
| то закрасить
все
```

При выполнении этой команды ЭВМ спросит Робота "сверху свободно?" Если Робот ответит **да**, то ЭВМ скомандует Роботу "закрасить". Если же Робот ответит **нет**, то ЭВМ вызов команды "закрасить" пропустит.

```
алг разметка выходов из коридора                    (А38)
дано |Робот в левой клетке горизонтального коридора (рис. 53)
надо |Робот вышел из коридора вправо, клетки коридора,
     |из которых есть выход вверх, закрашены
```



```
нач
| нц пока снизу стена
| | если сверху свободно
| | | то закрасить
| | все
| | вправо
| кц
кон
```

10.2. ОБЩИЙ ВИД КОМАНДЫ ЕСЛИ

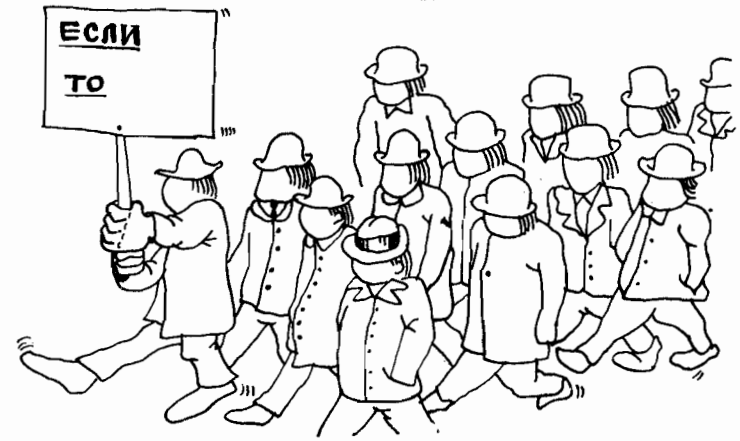
Общий вид команды **если** таков:

```
если условие                    если условие
| то серия 1          или       | то серия 1
| иначе серия 2                 все
все
```

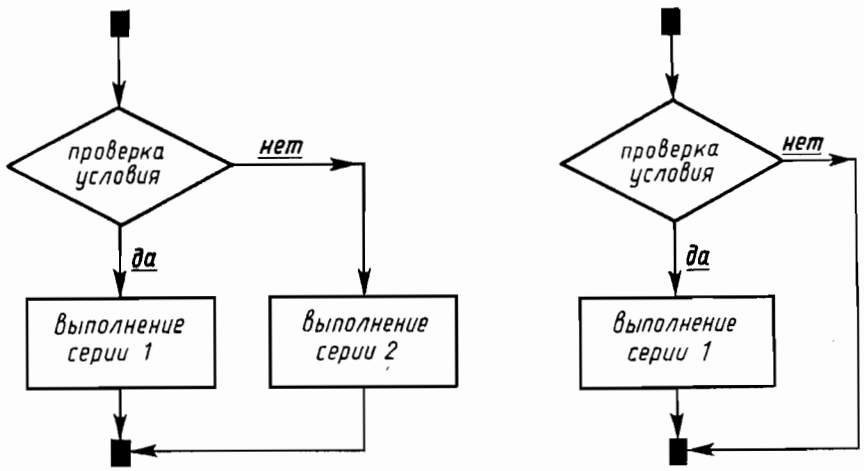
Служебные слова **если**, **то**, **иначе** имеют обычный смысл. Слово **все** означает конец команды. Это слово пишется строго под словом **если** и соединяется с ним вертикальной чертой. Между **то** и **иначе** — в одной или нескольких строках — записывается последовательность команд алгоритмического языка (серия 1). Между **иначе** и **все** записывается другая последовательность команд (серия 2). Серия 2 вместе со служебным словом **иначе** может отсутствовать.

При выполнении команды **если** ЭВМ сначала проверяет условие, записанное между **если** и **то** (задает Роботу вопрос). В результате проверки получается либо **да**, либо **нет**. Если получилось **да**, то выполняется серия 1, а если **нет** — то серия 2 (если она есть).

Если условие не соблюдается (Робот ответил **нет**), а серия 2 вместе с **иначе** отсутствует, то ЭВМ сразу переходит к выполнению команд, записанных после слова **все**.



10.3. ГРАФИЧЕСКАЯ СХЕМА ВЫПОЛНЕНИЯ КОМАНДЫ ЕСЛИ



10.4. ВТОРОЙ ПРИМЕР ИСПОЛЬЗОВАНИЯ КОМАНДЫ ЕСЛИ

```
алг сдвиг (арг цел m, n)                                    (A 39)
 дано |на поле Робота стен нет
 надо |Робот сместился на "вектор" (m, n) от исходного положения
нач
| если m>0
| | то вправо на (m)
| | иначе влево на (−m)
| все
| если n>0
| | то вверх на (n)
| | иначе вниз на (−n)
| все
кон
```

10.5. ТРЕТИЙ ПРИМЕР ИСПОЛЬЗОВАНИЯ КОМАНДЫ ЕСЛИ — РАЗМЕТКА ОПАСНЫХ КЛЕТОК КОРИДОРА

Робот стоит в левом конце горизонтального коридора. Требуется разметить (закрасить) те клетки коридора, в которых уровень радиации превышает предельно допустимое значение a.

```
алг разметка опасных клеток (арг вещ a)                     (A40)
 дано |Робот стоит в левой клетке горизонтального коридора
 надо |Робот вышел из коридора вправо, клетки, в которых
      |уровень радиации выше a, закрашены
```



```
нач
| нц пока снизу стена
| | если радиация>a
| | | то закрасить
| | все
| | вправо
| кц
кон
```

Здесь в команде **если** записано условие "радиация>a". При проверке этого условия ЭВМ вызовет команду "радиация", по которой Робот сообщит ЭВМ уровень радиации в той клетке, где он находится. ЭВМ проанализирует полученное число и, если оно больше a, вызовет команду "закрасить".

10.6. УСЛОВИЯ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Мы видели, что в командах **пока** и **если** в условия могут входить как команды-вопросы ("снизу стена", "клетка закрашена"), так и отношения между числами ("радиация>a"). Условия такого вида называют ***простыми***. Вот еще несколько примеров простых условий: "$x \neq 1$", "$(a+b)*(a-b)=0$", "$0 \leqslant x \leqslant 1$", "abs (x) + abs (y) $\leqslant$ R".

Из простых условий можно получать ***составные***, используя служебные слова **и**, **или**, **не** и скобки для указания порядка проверки условий. Вот несколько примеров простых и составных условий:

а) сверху свободно **или** слева свободно;
б) сверху свободно **и** снизу стена;
в) клетка закрашена **и** слева свободно;
г) слева стена **и** температура>0;
д) слева стена **и** (температура>100 **или** радиация>1);
е) слева стена **и** 0<температура<100;
ж) $x>0$ **и** $y>0$;
з) $x*x+y*y \leqslant 1$ **и** $x \geqslant 0$ **и** $y \geqslant 0$;
и) ($x>1$ **и** $y>1$) **или** ($x<1$ **и** $y<1$);
к) $b*b-4*a*c \geqslant 0$;
л) $y=x$;
м) $a>0$ **и** $b>0$ **и** $c>0$ **и** $a+b>c$;
н) $0 \leqslant t \leqslant 1$;
о) $a<b<c<d$;
п) **не** ($x=0$ **и** $y=0$).

Результатом проверки любого условия является либо **да** (если условие соблюдается), либо **нет** (если условие не соблюдается). Условие вида "А **или** Б" соблюдается (равно **да**), если соблюдается либо А, либо Б, либо они оба.

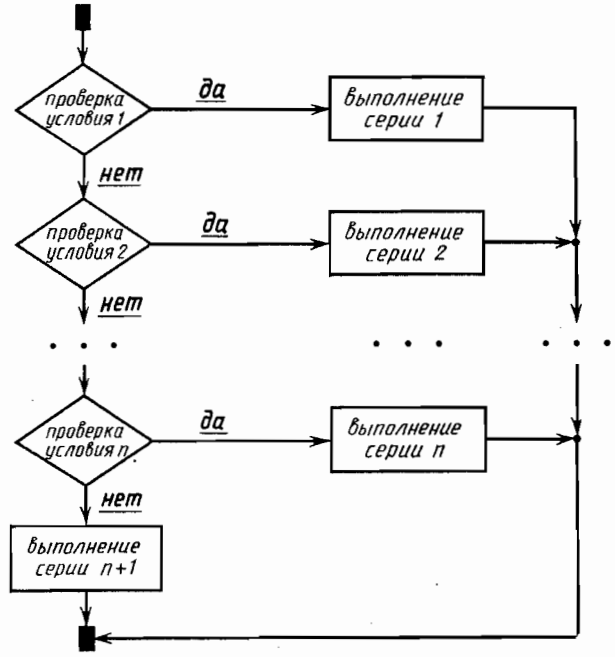
10.7. КОМАНДА ВЫБОР

В команде **если** указывается не более двух вариантов действий ЭВМ. Если нужно выполнить один из многих вариантов, то используется команда **выбор**:

```
выбор
| при условие 1:серия 1
| при условие 2:серия 2
| ...
| при условие n:серия n
| иначе серия n+1
все
```

Как и в команде **если**, серия n+1 вместе со служебным словом **иначе** может отсутствовать.

10.8. ГРАФИЧЕСКАЯ СХЕМА ВЫПОЛНЕНИЯ КОМАНДЫ ВЫБОР



10.9. ПРИМЕР АЛГОРИТМА С КОМАНДОЙ ВЫБОР

```
алг уйти из клетки                                   (А 41)
 дано |Робот в клетке, у которой хотя бы одной стены нет
 надо |Робот вышел из исходной клетки
```



```
нач
| выбор
| | при сверху свободно :вверх
| | при справа свободно:вправо
| | при снизу свободно  :вниз
| | при слева свободно  :влево
| все
кон
```

Здесь в команде выбора записаны четыре условия и четыре действия. Соблюдаться может сразу несколько условий, но действие будет выполнено только одно. Например, если со всех сторон свободно, то выполнена будет только команда "вверх".

10.10. КОМАНДЫ КОНТРОЛЯ

Вспомните: когда вы объясняете, как пройти к определенному месту, то наверняка говорите что-нибудь вроде "заверните за угол, там будет видна булочная". Это значит, что после выполнения команды "заверните за угол" надо проверить условие "видна булочная". Для записи таких контрольных условий в алгоритмическом языке есть специальная команда контроля

утв условие

После служебного слова **утв** (утверждение) записывается контрольное условие. Если при выполнении команды **утв** контрольное условие оказывается нарушенным (равно **нет**), ЭВМ прекращает выполнение алгоритма и сообщает, что возник отказ. Если же условие равно **да**, то выполнение алгоритма нормально продолжается так, как если бы команды **утв** не было вовсе.

Контрольные условия можно указать не только в команде **утв**, но и после **дано** и **надо**. Условие после **дано** проверяется в начале выполнения алгоритма, а условие после **надо** — в конце.

Вместо контрольного условия, которое проверяет ЭВМ, в команде **утв** (так же как и в **дано**/**надо**) можно записать комментарий, предназначенный только для человека и облегчающий понимание алгоритма.

10.11. ПРИМЕР АЛГОРИТМА С КОМАНДОЙ УТВ

Дано, что Робот стоит в левой клетке горизонтального коридора, от которого вверх отходят тупики размером в одну клетку (рис. 54). Требуется вывести Робота из коридора вправо (в клетку Б), а тупики закрасить (рис. 54).

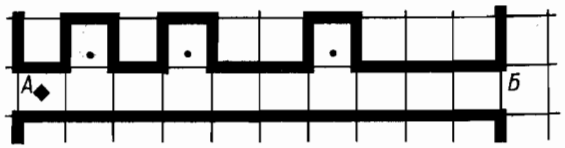


Рис. 54

Для решения этой задачи мы можем составить следующий алгоритм:

```
алг закрасить тупики                                        (А 42)
 дано |Робот стоит в левой клетке горизонтального коридора,
      |выше некоторых клеток коридора есть тупики размером
      |в одну клетку (рис. 54)
 надо |Робот вышел из коридора вправо, клетки всех тупиков
      |закрашены
нач
| нц пока снизу стена
| | если сверху свободно
| | | то
| | |  вверх
| | |   утв |Робот в тупике
| | |  закрасить
| | |  вниз
| | все
| | вправо
| кц
кон
```

В этом алгоритме команда **утв** содержит только комментарий, поясняющий алгоритм. Мы можем, однако, заменить ее на команду

утв слева стена **и** сверху стена **и** справа стена

В этом случае при выполнении алгоритма ЭВМ будет проверять указанное условие и попытка выполнить алгоритм в неподходящей обстановке (т. е. в обстановке, отличающейся от рисунка 54) приведет к отказу.

УПРАЖНЕНИЯ

1. Составьте алгоритмы со следующим заголовком:

```
алг разметка горячих клеток коридора
 дано |Робот стоит в левой клетке горизонтального коридора
 надо |Робот вышел из коридора вправо, клетки, в которых
      |температура выше 100 градусов, закрашены
```


2. Запишите на алгоритмическом языке условие, которому удовлетворяют точки, изображенные на рисунке 55.

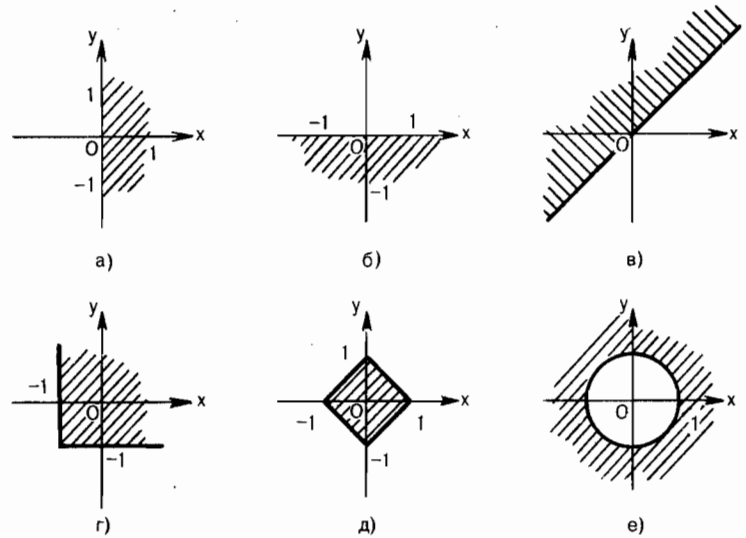


Рис. 55

3. Переделайте алгоритм А 40, считая опасными те клетки, в которых либо уровень радиации, либо температура превосходят заданные предельно допустимые значения.

4. На поле Робота стен нет. В ряду из десяти клеток правее Робота некоторые клетки закрашены. Составьте алгоритм, который закрашивает клетки

а) ниже каждой закрашенной;
б) выше и ниже каждой закрашенной;
в) левее каждой закрашенной;
г) правее каждой закрашенной;
д) левее и правее каждой закрашенной.

5. Воспользовавшись вспомогательными алгоритмами "вверх до стены", "вниз до стены", "вправо до стены", "влево до стены", составьте алгоритмы со следующими заголовками:

а)
```
алг переход в противоположный угол
 дано |Робот стоит в каком-то углу прямоугольника,
      |огороженного стенами. Других стен нет
 надо |Робот в противоположном углу
```

б)
```
алг к противоположной стене
 дано |Робот стоит у стены (но не в углу) внутри прямо-
      |угольника, обнесенного со всех сторон стенами;
      |внутри прямоугольника других стен нет
 надо |Робот перешел к противоположной стене
```

6. Робот внутри коридора без боковых выходов, идущего в неизвестном направлении. Составьте алгоритм, выводящий Робота из коридора.

7. Робот внутри тупика неизвестного направления. Составьте алгоритм, выводящий Робота из тупика.

8. Робот стоит на перекрестке, от которого отходят один коридор (без боковых выходов) и три тупика. Составьте алгоритм, после выполнения которого Робот окажется с противоположной стороны коридора.

§ 11. ВЕЛИЧИНЫ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ. КОМАНДА ПРИСВАИВАНИЯ

11.1. НЕОБХОДИМОСТЬ РАБОТЫ С ВЕЛИЧИНАМИ В ПРОЦЕССЕ ВЫПОЛНЕНИЯ АЛГОРИТМА

Рассмотрим следующую задачу. Робот расположен в клетке над горизонтальной стеной (рис. 56) неизвестной длины. Надо переместить Робота на клетку вниз — "сквозь стену" (рис. 56).

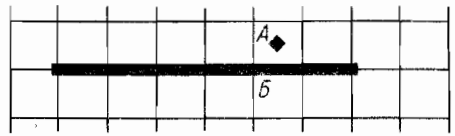


Рис. 56

Поскольку Робот сквозь стену проходить не умеет, стену нужно обойти. Сделать это можно так. Пока ниже Робота стена, будем двигать его вправо и по дороге считать число шагов (число команд "вправо"). Как только стена кончится, сместим Робота вниз и начнем двигать его обратно (влево). Тут нам и понадобится запомненная ранее информация о числе шагов вправо: скомандуем Роботу "влево" столько же раз, сколько было сделано шагов вправо, и он окажется в точке Б.

Если мы командуем Роботом сами, то провести этот план в жизнь не представляет труда. Но чтобы составить соответствующий алгоритм для ЭВМ, надо уметь запоминать, изменять и использовать информацию в памяти ЭВМ (в данном случае — информацию о числе сделанных вправо шагов). Для этого в алгоритмическом языке используются так называемые ***величины***.

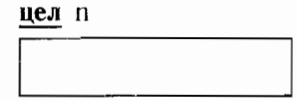
Термин "величина" заимствован из математики и физики, поскольку в алгоритмах для решения математических или физических задач величины алгоритмического языка соответствуют математическим или физическим величинам.

11.2. ИМЯ, ЗНАЧЕНИЕ И ТИП ВЕЛИЧИНЫ

Каждая величина имеет ***имя***, ***значение*** и ***тип***. ***Имя*** величины (например, "n", "x", "d") служит для обозначения величины в алгоритме. Во время выполнения алгоритма в каждый конкретный момент величина имеет какое-то ***значение*** (например, 22 или —107) либо ***не определена***. Если значением величины может быть только целое число, то величина называется ***целой*** или ***целочисленной***, если любое вещественное число — то ***вещественной***. Эта характеристика величины (в данном случае является ли величина целой или вещественной) называется ***типом*** величины. Позже мы познакомимся с величинами и других типов: логическими, символьными и др.

11.3. МОДЕЛЬ ПАМЯТИ ЭВМ

Для запоминания информации в ЭВМ имеется ***память***. Напомним (см. 6.9), что память ЭВМ удобно представлять в виде классной доски, на которой можно записывать информацию, читать, стирать, записывать заново и т. д. Место, отводимое в памяти ЭВМ для каждой величины, удобно изображать в виде прямоугольника. Значение величины (если она определена) записывается внутри прямоугольника, а тип и имя указываются сверху:



Если величина не определена, то в прямоугольнике ничего не пишется.

11.4. ОПИСАНИЕ ВЕЛИЧИН

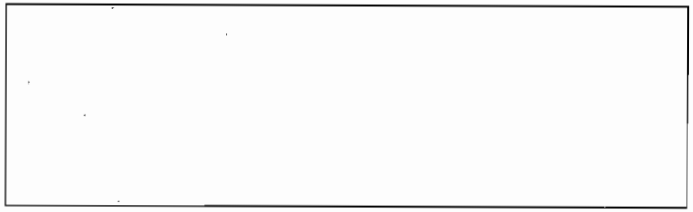
Для того чтобы ЭВМ могла работать с величиной, нужно указать тип и имя величины, например **цел** n. Такое указание называется ***описанием*** величины. Для величин, используемых в промежуточных вычислениях, описания располагаются в алгоритме сразу за словом **нач**:

```
алг вниз сквозь стену
  дано | Робот над горизонтальной стеной, других стен нет
  надо | Робот под стеной, на клетку ниже исходного положения
нач цел n
```

11.5. КАК ЭВМ ОТВОДИТ ВЕЛИЧИНЕ МЕСТО В ПАМЯТИ

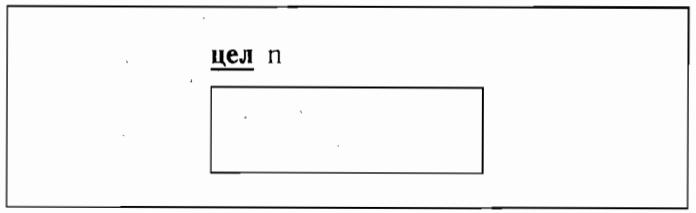
В начале выполнения алгоритма ЭВМ выделит для него часть памяти:

алг вниз сквозь стену



Встретив после слова **нач** описание "**цел** n", ЭВМ отведет внутри прямоугольника алгоритма место для хранения целочисленной величины n — величины с ***именем*** n ***типа*** **цел**:

алг вниз сквозь стену



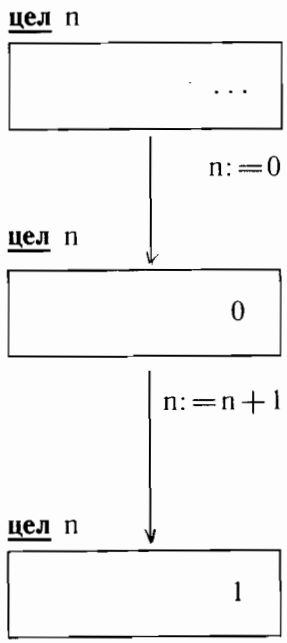
11.6. КОМАНДА ПРИСВАИВАНИЯ

Для того чтобы запомнить или изменить значение величины, в алгоритмическом языке есть специальная команда — ***команда присваивания***, которая записывается в виде:

имя величины := выражение

Знак ":=" (двоеточие, а потом равенство) называется знаком ***присваивания*** и читается как "присвоить" (например, команда "n:=e" читается "n присвоить e"). При выполнении команды присваивания ЭВМ сначала вычисляет записанное в правой части выражение (заменяя имена величин на их значения), а потом полученное значение выражения записывает в память.

11.7. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ КОМАНДЫ ПРИСВАИВАНИЯ



При выполнении команды присваивания "n:=0" ЭВМ запишет значение выражения (т. е. 0) внутрь прямоугольника величины "n" (старое значение, каково бы оно ни было, будет затерто).

Таким образом, после выполнения команды "n:=0" величина n будет иметь значение 0.

Если теперь выполнить команду "n:=n+1", то ЭВМ сначала вычислит значение выражения "n+1", т. е. заменит имя n на значение 0 и вычислит 0+1=1. После этого ЭВМ сотрет старое значение величины n и запишет новое (1).

Другими словами, после выполнения команды "n:=n+1" значение величины n будет увеличено на 1.

11.8. ПРИМЕР АЛГОРИТМА, РАБОТАЮЩЕГО С ВЕЛИЧИНАМИ

Запишем теперь алгоритм "вниз сквозь стену", используя для подсчета числа сделанных вправо шагов целочисленную величину n:

```
алг вниз сквозь стену                                    (А 43)
 дано |Робот над горизонтальной стеной, других стен нет
 надо |Робот под стеной, на клетку ниже исходного положения
нач цел n
| n:=0
| нц пока снизу стена
| | вправо; n:=n+1
| кц
|   утв |Робот вышел за край стены,
|       |n=число сделанных вправо шагов
| вниз
| влево на (n)
кон
```

При выполнении этого алгоритма ЭВМ сначала присвоит величине n в своей памяти значение 0 (команда "n:=0"). Затем в цикле ЭВМ будет командовать Роботу "вправо" и тут же увели-

чивать значение n на 1 (команда "n: =n+1"). После каждого выполнения тела цикла значение n будет равно числу сделанных Роботом шагов вправо. После окончания цикла значением n будет общее число сделанных вправо шагов. Поэтому, когда при выполнении вспомогательного алгоритма "влево на(n)" ЭВМ n раз скомандует "влево", Робот окажется в точности на клетку ниже исходного положения (рис. 57).

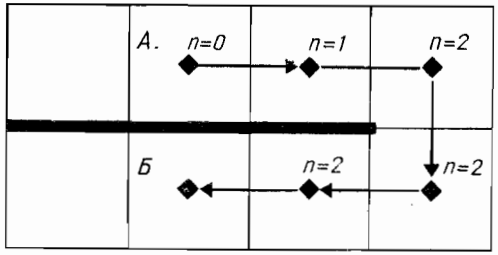


Рис. 57

11.9. ЕЩЕ ОДИН ПРИМЕР ИСПОЛЬЗОВАНИЯ ВЕЛИЧИН ДЛЯ ЗАПОМИНАНИЯ ИНФОРМАЦИИ

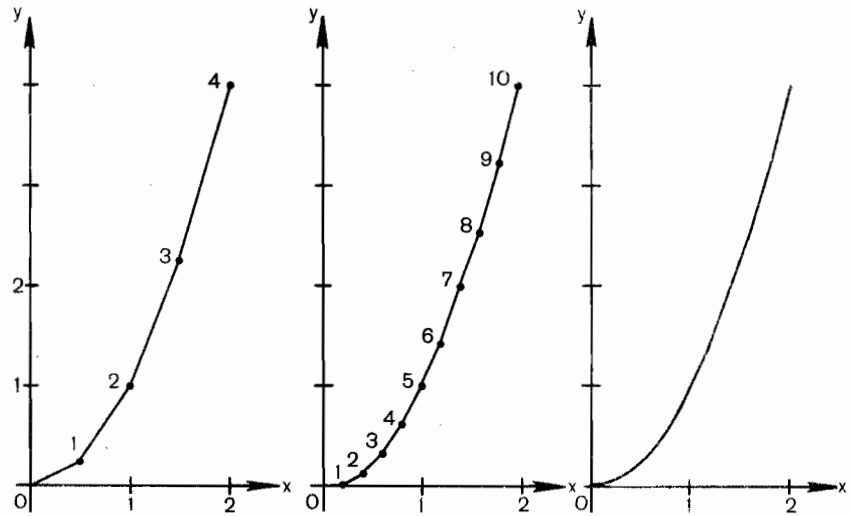
```
алг закрасить радиоактивные клетки                    (А 44)
 дано |Робот стоит левее горизонтального коридора
 надо |закрашены все клетки коридора, в которых уровень
      |радиации выше, чем в исходном положении Робота.
      |Робот вышел из коридора вправо
нач вещ x
| x:=радиация |запоминание уровня радиации в начальной клетке
| вправо      |вход в коридор
| нц пока снизу стена |т. е. пока Робот в коридоре
| | если радиация>x
| | | то закрасить
| | все
| | вправо                |переход в следующую клетку
| кц
кон
```

Здесь вещественная величина x используется для хранения информации об уровне радиации в начальной клетке (исходном положении Робота).

11.10. РИСОВАНИЕ ПАРАБОЛЫ

Рассмотрим следующую задачу: изобразить с помощью Чертежника график функции (параболу) $y=x^2$. Условие этой задачи

нуждается в уточнении. Во-первых, нарисовать весь график нельзя — ведь он бесконечен. Поэтому будем рисовать только часть графика, например от $x=0$ до $x=2$. Во-вторых, Чертежник не умеет рисовать ничего, кроме отрезков, а график функции $y=x^2$ — это кривая. Поэтому параболу придется заменить ломаной. Если вершины ломаной лежат на параболе и звенья ломаной очень короткие, то ломаная на вид почти не отличается от параболы. На рисунке 58, *а* ломаная имеет 4 звена, на 58, *б* — 10 звеньев, на 58, *в* — 100 звеньев.



а) 4 звена б) 10 звеньев в) 100 звеньев

Рис. 58

В общем случае с помощью Чертежника можно нарисовать параболу на некотором участке (от a до b) в виде ломаной из какого-то числа звеньев (n). Будем рисовать ломаную слева направо, начиная от точки (a, a ** 2) и кончая точкой (b, b ** 2). От вершины к вершине будем перемещаться с помощью команды Чертежника "сместиться в точку".

Используем вещественную величину x для запоминания x-координаты очередной вершины ломаной. Разобьем отрезок от a до b на n равных частей длины $d=(b-a)/n$ (рис. 59).

При рисовании очередного звена ломаной величину x надо просто увеличивать на d — это можно сделать с помощью команды присваивания "x: =x+d":

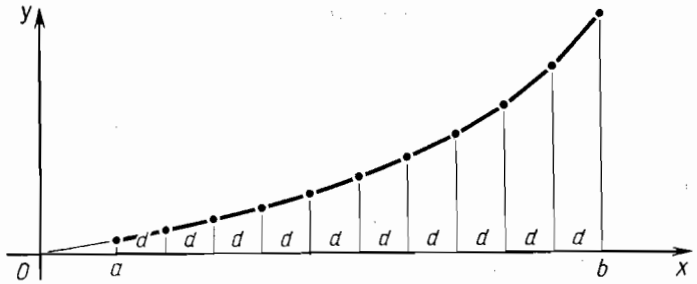


Рис. 59

```
алг парабола (арг вещ a, b, арг цел n)                    (A 45)
 дано | n>0, перо Чертежника поднято
 надо | нарисован график функции y=x ** 2 на участке от a до b в
      | виде ломаной из n звеньев; перо в точке (b, b ** 2)
      | и поднято
нач вещ x, d
| x:=a; d:=(b-a)/n
| сместиться в точку (a, a ** 2)
| опустить перо
| нц n раз
| | x:=x+d
| | сместиться в точку (x, x ** 2)
| кц
| поднять перо
кон
```

При выполнении вызова "парабола (0, 2, 10)" ЭВМ выдаст Чертежнику последовательно 13 команд

сместиться в точку (0, 0)
опустить перо
сместиться в точку (0.2, 0.04)
сместиться в точку (0.4, 0.16)
сместиться в точку (0.6, 0.36)
сместиться в точку (0.8, 0.64)
сместиться в точку (1.0, 1.00)
сместиться в точку (1.2, 1.44)
сместиться в точку (1.4, 1.96)
сместиться в точку (1.6, 2.56)
сместиться в точку (1.8, 3.24)
сместиться в точку (2.0, 4.00)
поднять перо

и Чертежник нарисует параболу на участке от 0 до 2 в виде ломаной из 10 звеньев (рис. 58, *б*).

УПРАЖНЕНИЯ

1. Значение величины x равно 3. Чему оно будет равно после выполнения команды:

а) x:=5; б) x:=x+5; в) y:=x?

2. После выполнения команды

а) x:=x+5; б) x:=−x; в) y:=x

или серии команд

г) y:=1; x:=x+y; д) y:=x; x:=y

значение величины x стало равно 3. Чему было равно значение величины x до выполнения в каждом из этих случаев?

3. После выполнения команды присваивания "x:=x+y" значение величины x равно 3, а значение y равно 5. Чему были равны значения величин x и y до выполнения команды?

4. Значение величины x равно a, значение y равно b. После выполнения каких из указанных ниже последовательностей команд значения величин x и y поменяются, т. е. x станет равно b, а y станет равно a?

а)	б)	в)	г)
x:=y y:=x	t:=x x:=y y:=t	x:=x+y y:=x−y x:=x−y	t:=x y:=t x:=y

5. Опишите, что произойдет при выполнении алгоритма "вниз сквозь стену" для начальных положений Робота, указанных на рисунке 60.

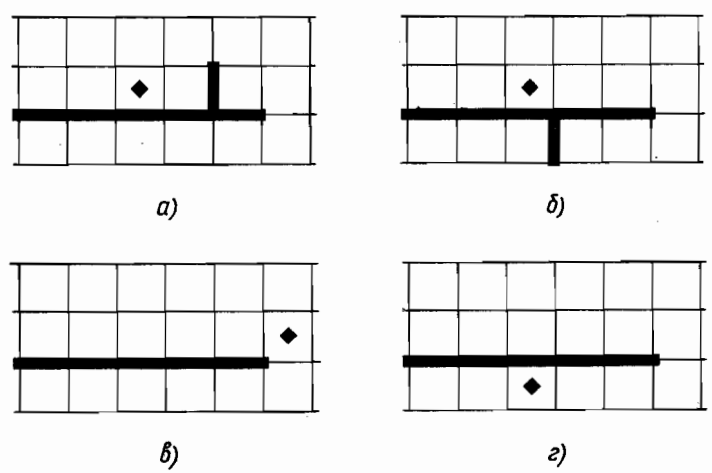


Рис. 60

6. Дан алгоритм:

```
алг график (арг вещ b, арг цел n)                    (А 46)
 дано  n>0 |перо Чертежника поднято
 надо  |нарисован график некоторой функции на участке от
       |0 до b в виде ломаной из n звеньев; перо в конце
       |графика и поднято
нач вещ x, d, v
| x:=0; d:=b/n
| сместиться в точку (0, 0)
| опустить перо
| нц n раз
| | v:=2*x*d+d*d; x:=x+d
| | сместиться на вектор (d, v)
| кц
| поднять перо
кон
```

а) Как будет выглядеть память ЭВМ при выполнении вызова "график (0, 10)"?

б) Что нарисует Чертежник при выполнении этого вызова?

7. Дано, что Робот находится перед входом в тупик, идущий вправо. Составьте алгоритм, при выполнении которого с помощью Чертежника будет нарисован график радиоактивности в тупике (считая, что единица масштаба по оси *Ox* равна размеру клетки на поле Робота).

8. Составьте алгоритмы со следующими заголовками:

а) **алг** вниз до стены закрасить и вернуться
 дано |где-то ниже Робота есть стена
 надо |Робот дошел до этой стены, закрасил клетку
 |и вернулся в исходное положение

б) **алг** закрасить клетку в правом нижнем углу
 дано |Робот где-то внутри прямоугольника, огороженно-
 |го стенами, других стен нет
 надо |Робот закрасил клетку в правом нижнем углу
 |прямоугольника и вернулся в исходное положение

в) **алг** отойти вдвое дальше от левой стены
 дано |где-то левее Робота есть стена, других стен нет
 надо |Робот отошел вправо на расстояние от стены
 |вдвое большее, чем исходное

г) **алг** симметрия
 дано |где-то ниже Робота есть горизонтальная стена длиной
 |в одну клетку, других стен нет
 надо |Робот оказался в положении, симметричном исходному
 |относительно стены

д) **алг** симметрия
 дано |где-то ниже Робота есть горизонтальная стена, которая
 |и вправо и влево кончается, других стен нет
 надо |Робот оказался в положении, симметричном исходному
 |относительно стены

е) **алг** обойти прямоугольное препятствие
 дано |Робот над прямоугольником, огороженным стенами,
 |других стен нет
 надо |Робот под прямоугольником на той же вертикали

ж*) **алг** вниз сквозь бесконечную стену
 дано |Робот над горизонтальной стеной, в одну сторону
 |уходящей в бесконечность, других стен нет
 надо |Робот под стеной на клетку ниже исходного положения

9. Робот находится в левом верхнем углу прямоугольника, огороженного стенами. Составьте алгоритм, после выполнения которого Робот будет стоять в клетке с минимальным уровнем радиации.

§ 12. РЕЗУЛЬТАТЫ АЛГОРИТМОВ И АЛГОРИТМЫ-ФУНКЦИИ

12.1. ВИДЫ ВЕЛИЧИН В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

В алгоритмическом языке используется несколько ***видов величин***. В алгоритме "парабола" (А 45) предыдущего параграфа величины x и d участвуют в промежуточных вычислениях и называются ***промежуточными***. Величины a, b и n этого алгоритма, как мы уже знаем, называются ***аргументами*** и служат для передачи информации от основного алгоритма вспомогательному.

В этом параграфе мы изучим ***величины-результаты*** (или просто ***результаты***), которые служат для передачи информации от вспомогательного алгоритма основному.

12.2. ПРОСТЕЙШИЙ ПРИМЕР АЛГОРИТМА С РЕЗУЛЬТАТАМИ

```
алг гипотенуза (арг вещ a, b, рез вещ c)             (А 47)
 дано  a≥0 и b≥0 |длины катетов треугольника
 надо  |c=длина гипотенузы этого треугольника
нач
| c:=sqrt (a ** 2+b ** 2)
кон
```

Запись **рез вещ** с означает, что результат (**рез**) выполнения этого алгоритма — одна вещественная (**вещ**) величина с.

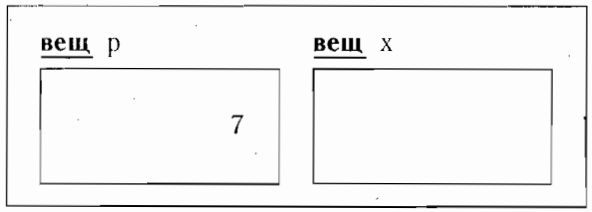
12.3. ВЫПОЛНЕНИЕ АЛГОРИТМА С РЕЗУЛЬТАТАМИ

При вызове вспомогательного алгоритма на месте его результатов указываются имена величин основного алгоритма, в которых должны оказаться вычисленные значения. Например, если нам надо в основном алгоритме "вычисление" найти гипотенузу x прямоугольного треугольника с катетами p−1 и p+1, то достаточно написать вызов вспомогательного алгоритма "гипотенуза" (A47):

```
алг вычисление
нач вещ p, x
| . . .
| гипотенуза (p−1, p+1, x)
| . . .
кон
```

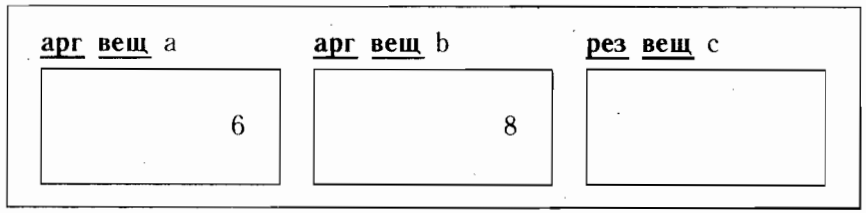
Пусть к моменту выполнения этого вызова величина p имеет значение 7:

алг вычисление



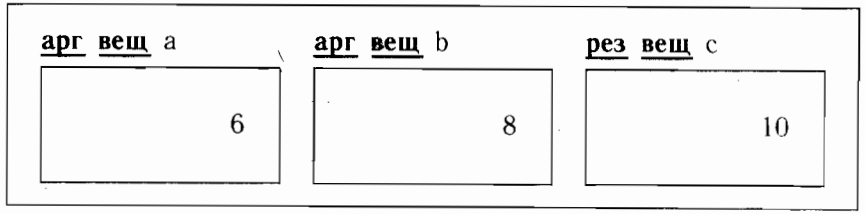
Встретив вызов "гипотенуза (p−1, p+1, x)", ЭВМ начнет выполнять вспомогательный алгоритм "гипотенуза", отведет часть памяти для аргументов a, b и результата c и запомнит начальные значения аргументов: a=p−1=6 и b=p+1=8:

алг гипотенуза



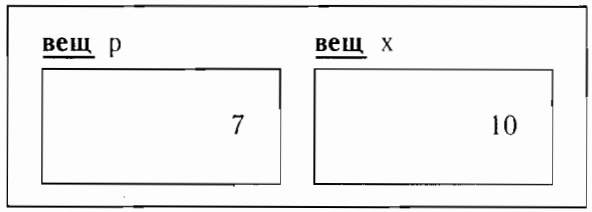
В памяти ЭВМ при этом будет одновременно храниться информация, относящаяся и к алгоритму "вычисление", и к алгоритму "гипотенуза". Далее ЭВМ выполнит алгоритм "гипотенуза" и вычислит значение величины-результата c:

алг гипотенуза



Встретив строчку кон, ЭВМ скопирует значение результата c в величину x основного алгоритма "вычисление", указанную в вызове "гипотенуза (p−1, p+1, x)":

алг вычисление



После этого ЭВМ закончит выполнение вспомогательного алгоритма "гипотенуза", сотрет из памяти весь прямоугольник алгоритма "гипотенуза" (со всем его содержимым) и продолжит выполнение основного алгоритма "вычисление".

12.4. ОБЩИЕ ПРАВИЛА ВЫПОЛНЕНИЯ КОМАНДЫ ВЫЗОВА ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА

1. Перед началом выполнения вспомогательного алгоритма ЭВМ отводит для него место в памяти и устанавливает значения аргументов, указанные в команде вызова в основном алгоритме.
2. На время выполнения вспомогательного алгоритма выполнение основного алгоритма приостанавливается.
3. В конце выполнения вспомогательного алгоритма значения его результатов присваиваются величинам, указанным в команде вызова в основном алгоритме.
4. После окончания вспомогательного алгоритма все, что с ним связано, стирается из памяти ЭВМ. (Если алгоритм вызывается еще раз, то все начинается сначала: ЭВМ снова отводит место в памяти, присваивает значения аргументам и т. д.)

12.5. РЕШЕНИЕ КВАДРАТНОГО УРАВНЕНИЯ

```
алг квур (арг вещ a, b, c, рез цел n, рез вещ x1, x2)          (A48)
дано a≠0 |a, b, c — коэффициенты уравнения ax²+bx+c=0
надо |n — число разных корней, x1, x2 — корни (если есть)
```

```
нач вещ d
| d:=b**2-4*a*c |дискриминант
| если d<0
| | то n:=0
| | иначе
| | если d=0
| | | то n:=1; x1:=-b/(2*a); x2:=x1
| | | иначе
| | | n:=2
| | | x1:=(-b-sqrt(d))/(2*a)
| | | x2:=(-b+sqrt (d))/(2*a)
| | все
| все
кон
```

Этот алгоритм можно записать короче, если заметить, что число корней квадратного уравнения равно sign (d)+1:

если d<0, то sign (d) = −1 и sign (d)+1=0;
если d=0, то sign (d) = 0 и sign (d)+1=1;
если d>0, то sign (d) = +1 и sign (d)+1=2.

```
алг квур (арг вещ a, b, c, рез цел n, рез вещ x1, x2)        (A49)
 дано a≠0 |a, b, c — коэффициенты квадратного уравнения
          |ax^2+bx+c=0
 надо |n — число разных корней, x1, x2 — корни (если есть)
нач вещ d
| d:=b**2-4*a*c |дискриминант
| n:=sign(d)+1  |число корней уравнения
| если n>0 то
| | x1:=(-b-sqrt(d))/(2*a)
| | x2:=(-b+sqrt(d))/(2*a)
| все
кон
```

12.6. ИНФОРМАЦИОННЫЕ АЛГОРИТМЫ

Здесь мы впервые встретились с алгоритмами, которые предназначены для "чистой" переработки информации — алгоритмы "гипотенуза" и "квур" никакими исполнителями не управляют, а лишь перерабатывают значения аргументов в значения результатов. Такие алгоритмы принято называть чисто информационными или вычислительными. Приведем еще один пример такого алгоритма:

```
алг сумма вклада (арг вещ a, арг цел n, рез вещ s)          (A50)
 дано |в начале каждого года в течение n лет на срочный вклад
      |в сберкассе вносится сумма в размере a рублей
 надо |s — сумма вклада по окончании n лет
нач
| s:=0 |сумма вклада до первого поступления денег
| нц n раз
| | s:=s+a    |поступление a рублей в начале каждого года
| | s:=1.03*s |начисление процентов (3%) в конце каждого
| |           |года
| кц
кон
```


12.7. АЛГОРИТМ С РЕЗУЛЬТАТАМИ ПРИ УПРАВЛЕНИИ РОБОТОМ

```
алг средний уровень радиации в коридоре (рез вещ r)          (A51)
 дано |Робот в левой клетке коридора, уходящего вправо
 надо |r=средний уровень радиации в коридоре,
      |Робот вышел из коридора вправо
нач цел n, вещ R
| n:=0; R:=0
| нц пока снизу стена
| | n:=n+1; R:=R+радиация
| | вправо
| кц
| утв |n=длина коридора, R=суммарная радиация
| r:=R/n
кон
```

12.8. АЛГОРИТМЫ-ФУНКЦИИ

Если нам нужно использовать корень квадратный из числа 13, мы пишем sqrt(13). Если нам нужно значение синуса в точке π/7, мы пишем sin (π/7). Другими словами, если нам нужно использовать в арифметическом выражении алгоритмического языка значение одной из перечисленных на странице 56 функций, то мы просто пишем имя функции, а в скобках указываем аргументы.

В алгоритмическом языке предусмотрена возможность задания новых функций, которые можно использовать наравне со стандартными функциями sqrt, sin, cos и другими. Для этого надо написать ***алгоритм вычисления функции***, или коротко ***алгоритм-функцию.***

Алгоритм вычисления функции отличается от обычного алгоритма формой записи заголовка и использованием специального служебного слова **знач**. В остальном алгоритм вычисления функции записывается по обычным правилам.

12.9. ПРИМЕР АЛГОРИТМА-ФУНКЦИИ

Пусть, например, мы хотим использовать в основном алгоритме функцию $s(t)=\frac{t^2}{2}$, выражающую путь, пройденный телом

за время t с момента начала равноускоренного движения при ускорении a=1. Алгоритм вычисления этой функции записывается так:

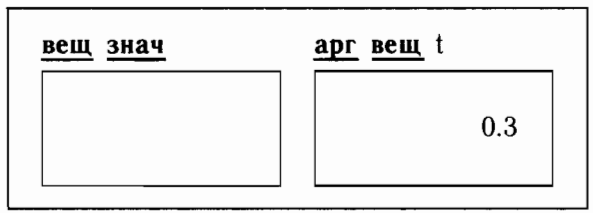
```
алг вещ s (арг вещ t)                                    (А52)
нач
| знач:=t**2/2
кон
```

Служебное слово **вещ** перед именем "s" алгоритма-функции показывает, что значением функции s является вещественное число. Внутри алгоритма для обозначения значения функции используется величина со служебным именем **знач**. Значение этой величины в момент окончания выполнения алгоритма и считается значением функции.

12.10. ВЫПОЛНЕНИЕ АЛГОРИТМА-ФУНКЦИИ

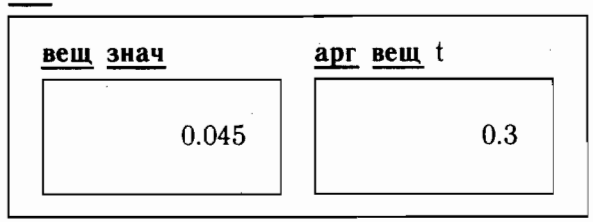
Рассмотрим работу ЭВМ при выполнении команды присваивания "y:=s(0.3)*1000". В правой части этой команды используется запись "s(0.3)", которая означает, что надо вызвать алгоритм-функцию s и вычислить ее значение для аргумента t=0.3. Встретив такой ***вызов функции***, ЭВМ отведет место в памяти для алгоритма-функции s и запомнит значение аргумента:

алг s



Тело алгоритма s состоит из одной команды: "**знач**:=t**2/2". В соответствии с правилами алгоритмического языка ЭВМ заменит в правой части имя аргумента t на его значение 0.3, вычислит выражение 0.3**2/2=0.09/2=0.045 и присвоит полученное значение величине **знач**:

алг s



На этом выполнение алгоритма-функции s закончится. Встретив строку **кон**, ЭВМ подставит полученное значение величины **знач** — число 0.045 — в формулу s(0.3)*1000 вместо вызова s(0.3) и сотрет из памяти прямоугольник "**алг** s" со всем его содержимым.

Далее ЭВМ вычислит выражение 0.045*1000=45 и присвоит полученное значение величине y. На этом выполнение команды присваивания "y:=s(0.3)*1000" закончится.

12.11. ПОСТРОЕНИЕ ГРАФИКА ПРОИЗВОЛЬНОЙ ФУНКЦИИ

Если заменить в алгоритме рисования параболы (А45) x**2 на f(x), можно получить алгоритм рисования графика функции f, заданной алгоритмом-функцией **алг вещ** f (**арг вещ** x):

```
алг график (арг вещ a, b, арг цел n)                     (А53)
 дано |n>0, перо Чертежника поднято
 надо |нарисован график функции y=f(x) на участке от a до b
      |в виде ломаной из n звеньев; перо в точке (b, f(b)) и
      |поднято
нач вещ x, d
| x:=a; d:=(b-a)/n
| сместиться в точку (a, f(a))
| опустить перо
| нц n раз
| | x:=x+d
| | сместиться в точку (x, f(x))
| кц
| поднять перо
кон
```

Например, для задания функции, изображенной на рисунке 61, *а*, можно составить следующий алгоритм:

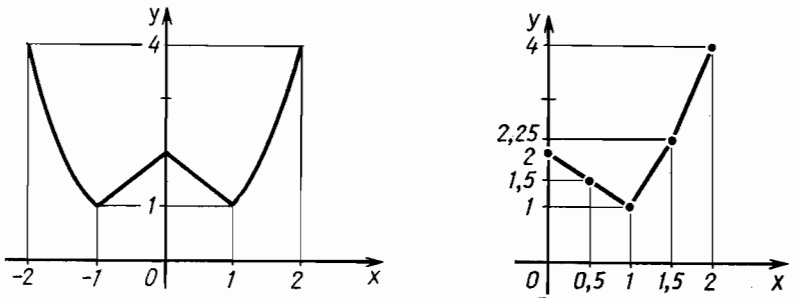
```
алг вещ f (арг вещ x)                                    (А54)
нач
| если -1≤x≤1
| | то знач:=2-abs(x)
| | иначе знач:=x**2
| все
кон
```

$$f(x)=\begin{cases}2-|x|, & \text{если } |x|\leqslant 1\\ x^{**}2, & \text{если } |x|>1\end{cases}$$

В этом случае при выполнении вызова "график (0, 2, 4)" ЭВМ 5 раз обратится к вспомогательному алгоритму f для вычисления значений функции f в точках 0.0, 0.5, 1.0, 1.5, 2.0 и, используя эти значения, выдаст Чертежнику 7 команд:

сместиться в точку (0, 2)
опустить перо
сместиться в точку (0.5, 1.5)
сместиться в точку (1.0, 1.0)
сместиться в точку (1.5, 2.25)
сместиться в точку (2.0, 4.0)
поднять перо

В результате будет нарисована ломаная, изображенная на рисунке 61, *б*. Поскольку в вызове указано, что ломаная должна иметь всего 4 звена, она значительно отличается от графика функции.



а) График функции *б) Ломаная из 4 звеньев*

Рис. 61

УПРАЖНЕНИЯ

1. Составьте алгоритмы со следующими заголовками:

а) **алг** расстояние до стены вправо (**рез цел** n)
дано |правее Робота есть стена
надо |положение Робота не изменилось, n=число шагов, |которое надо сделать Роботу, чтобы подойти к стене |вплотную (т. е. число клеток между Роботом и стеной)

б) **алг** расстояния вправо и влево (**рез цел** n1, n2)
дано |правее и левее Робота есть стены
надо |положение Робота не изменилось, n1=расстояние |вправо до стены, n2=расстояние влево до стены

в) **алг** к ближайшей стене вправо или влево
дано |правее и левее Робота есть стены
надо |Робот подошел к ближайшей из этих стен

г) **алг** подсчет числа опасных клеток (**рез цел** n)
дано |Робот на клетку левее тупика, уходящего вправо
надо |положение Робота не изменилось, n=число клеток, |в которых и температура и уровень радиации выше |средних значений в тупике

д) **алг** биквур (**арг вещ** p, q, **рез цел** n, **рез вещ** x1, x2, x3, x4)
дано |p и q — коэффициенты биквадратного уравнения |x**4+p*x**2+q=0
надо |n — число разных корней, x1 — x4 — корни (если есть)

2. Какие команды ЭВМ выдаст Чертежнику при выполнении вызова "график (0, 2, 10)" для алгоритмов А53 и А54?

3. Составьте алгоритм-функцию, который вычисляет:

а) максимальную высоту (в метрах) подъема тела, брошенного со скоростью v м/с вертикально вверх;

б) максимальную высоту (в метрах) подъема тела, брошенного со скоростью v м/с под углом α к горизонту;

в) количество максимальных среди трех чисел a, b, c;

г) количество разных среди трех чисел a, b, c;

д) среднее по величине среди трех чисел a, b, c;

е) расстояние от точки на числовой оси с координатой x до ближайшей точки отрезка [0, 1];

ж) расстояние от точки (x, y) на плоскости до ближайшей точки единичной окружности с центром в начале координат;

з) расстояние от точки (x, y) на плоскости до ближайшей точки отрезка [0, 1] на оси *Ox*;

и) число различных корней уравнения $(x-1)(x^2+bx+c)=0$;

к) число целых решений квадратного неравенства $n^2+n+a<0$;

л) наименьший делитель целого числа $n>0$, отличный от 1;

м) последнюю цифру в десятичной записи целого числа $n>0$;

н) количество цифр в десятичной записи целого числа $n>0$;

о) старшую цифру в десятичной записи целого числа $n>0$.

4. Выполните алгоритм (А55) при m0=14, n0=21:

```
алг цел НОД (арг цел m0, n0)                                (А55)
 дано m0>0 и n0>0
надо | знач=наибольший общий делитель m0 и n0
нач цел m, n
| m:=m0; n:=n0
| нц пока m≠n
| | если m>n
| | | то    m:=m-n
| | | иначе n:=n-m
| | все
| кц
|   утв m=n |и НОД (m, n)=НОД (m0, n0)
| знач :=m
кон
```

5. Используя алгоритм-функцию НОД (А55), составьте алгоритмы со следующими заголовками:

а) **алг цел** НОК (**арг цел** m0, n0)
   дано m0>0 **и** n0>0
   надо | **знач** = наименьшее общее кратное m0 и n0

б) **алг** сокращение дроби (**арг цел** a, b, **рез цел** a1, b1)
   дано a>0 **и** b>0 |числитель и знаменатель дроби
   надо |a1 и b1 — числитель и знаменатель сокращенной
   |дроби

6. Последовательность Фибоначчи строится так: первый и второй члены последовательности Фибоначчи равны 1, каждый следующий равен сумме двух предыдущих (1, 1, 2, 3, 5, 8, 13, 21, ...). Составьте алгоритм-функцию, который вычисляет:
   а) n-й член последовательности Фибоначчи;
   б) сумму первых n членов последовательности Фибоначчи.

7. Составьте алгоритмы-функции со следующими заголовками:

а) **алг вещ** площадь (**арг вещ** a, b, c)
   дано a≥0 **и** b≥0 **и** c≥0 |длины сторон треугольника
   надо |**знач**=площадь треугольника

б) **алг цел** число закрашенных клеток на север (**арг цел** n)
   дано n≥0 |севернее Робота стен нет
   надо |**знач** = число закрашенных клеток среди n клеток
   |к северу от Робота, Робот в исходном положении

в) **алг цел** расстояние до стены вниз
   дано |где-то ниже Робота есть стена
   надо |положение Робота не изменилось, **знач** =число шагов,
   |которое надо сделать Роботу, чтобы подойти к стене
   |вплотную (т. е. число клеток между Роботом и стеной)

г) **алг цел** число выходов из клетки
   надо |положение Робота не изменилось, **знач**=число сторон клетки, не перегороженных стенами

§ 13. КОМАНДЫ ВВОДА / ВЫВОДА ИНФОРМАЦИИ. ЦИКЛ ДЛЯ

13.1. КОМАНДЫ ВВОДА И ВЫВОДА ИНФОРМАЦИИ

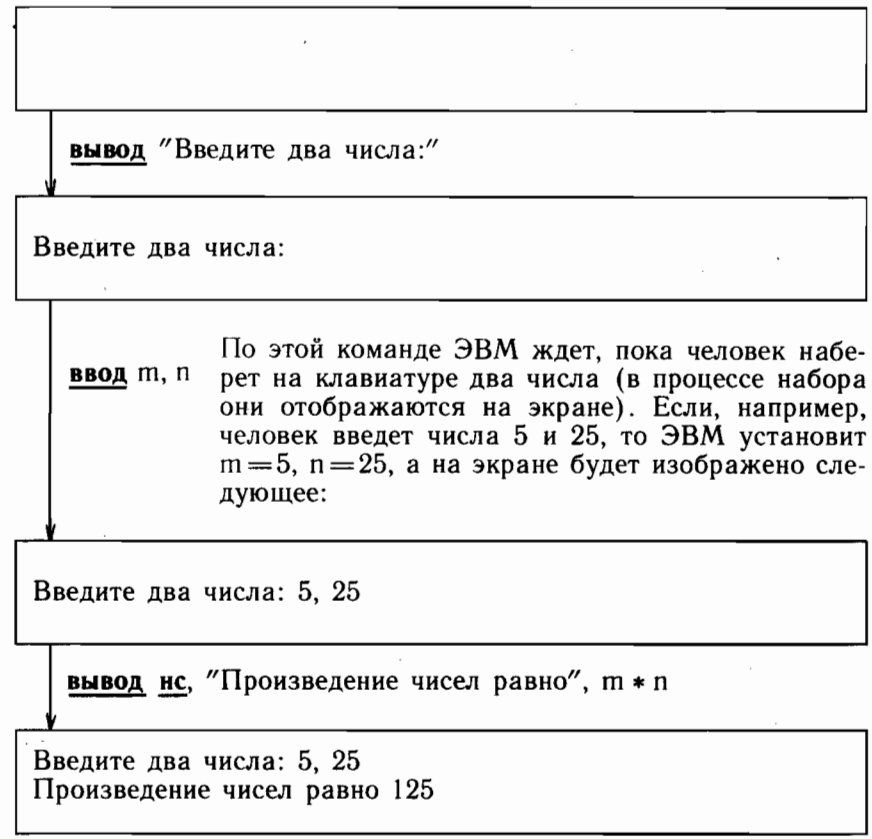
До сих пор, составляя алгоритмы, мы считали, что ЭВМ выполняет их автоматически, без какого-либо взаимодействия с человеком. Часто, однако, требуется организовать обмен информацией ("диалог") между человеком и ЭВМ в процессе выполнения алгоритма. Для этого в алгоритмическом языке есть специальные команды ***вывода*** информации из памяти ЭВМ на экран и ***ввода*** информации с клавиатуры (от человека) в память ЭВМ.

Поскольку в алгоритмическом языке для запоминания информации используются величины, то в командах ввода/вывода указываются имена величин, значения которых надо вывести (показать на экране) или ввести (запомнить в памяти ЭВМ).

13.2. ПРОСТЕЙШИЙ ПРИМЕР АЛГОРИТМА С КОМАНДАМИ ВВОДА/ВЫВОДА

```
алг произведение                                    (А56)
нач цел m, n
| вывод "Введите два числа:"
| ввод m, n
| вывод нс, "Произведение чисел равно", m*n
кон
```

При выполнении этого алгоритма экран будет меняться следующим образом:



Служебное слово **нс** (новая строка) указывает ЭВМ, что информация должна выводиться на новую строку.

13.3. РАБОТА КОМАНД ВВОД И ВЫВОД

В команде **ввод** через запятую указываются имена величин. При выполнении команды ЭВМ ждет, пока человек введет соответствующие значения, и по окончании ввода присваивает введенные значения указанным величинам. Признаком конца ввода служит нажатие на специальную клавишу (обычно на этой клавише изображена изогнутая стрелка "↵"). При вводе нескольких чисел они отделяются друг от друга запятой или пробелом.

В команде **вывод** можно через запятую указать тексты, имена величин и выражения. При выполнении команды ЭВМ вычислит значения выражений, подставит значения величин и изобразит на экране то, что получится (если информация не уместится в одной строке, она будет перенесена на следующую). Перейти на новую строку можно и до окончания текущей, написав в команде вывода служебное слово **нс** (новая строка).

Если в процессе вывода информация заполнит весь экран, то при переходе на следующую строку весь экран сдвинется вверх, вытеснив верхнюю строку "за экран". Таким образом, при продолжении вывода изображение побежит по экрану снизу вверх и в конце на экране останутся лишь последние из выведенных строк.

13.4. ЕЩЕ ОДИН ПРИМЕР

Вспомним задачу определения суммы вклада (А50). Составим еще один алгоритм для решения этой задачи, считая, что исходные данные (размер ежегодного вклада "а" и количество лет "n") надо запросить у человека (т. е. ввести с клавиатуры), а ответ — показать на экране. При этом воспользуемся составленным ранее алгоритмом А50 как вспомогательным:

```
алг вычисление суммы вклада                                    (А57)
нач вещ a, s, цел n
| вывод "Введите размер ежегодного вклада и число лет:"
| ввод a, n
| сумма вклада (a, n, s)
| вывод нс, "Через ", n," лет общая сумма будет", s
кон
```

Если при выполнении этого алгоритма ввести числа 600 и 10, то по его окончании экран будет таким:

```
Введите размер ежегодного вклада и число лет: 600, 10
Через 10 лет общая сумма будет 7084.677
```


13.5. ДИАЛОГОВЫЕ СИСТЕМЫ

В рассмотренных алгоритмах команды ввода/вывода использовались для ввода начальных данных и вывода результатов вычислений, т. е. по схеме "ввод информации — обработка — вывод". При такой схеме обработка информации производится автоматически, без участия человека.

Команды ввода/вывода могут быть использованы и для организации совместной работы человека и ЭВМ, когда решения принимает человек, а рутинную работу выполняет ЭВМ. Такие алгоритмы называются ***диалоговыми системами.***

13.6. ПРИМЕР АЛГОРИТМА С ЦИКЛОМ ДЛЯ

Пусть требуется составить алгоритм, который вводит целое число n, а затем изображает на экране квадраты первых n натуральных чисел (от 1 до n). Мы можем воспользоваться для решения этой задачи циклом **пока:**

```
алг квадраты                                                   (А58)
нач цел n, i
| вывод "Введите количество квадратов:"; ввод n
| i:=1
| нц пока i⩽n
| | вывод нс, i**2
| | i:=i+1
| кц
кон
```

Команда ввода "**вывод нс**, i**2" в этом алгоритме будет выполнена сначала для i=1, затем для i=2, затем для i=3 и т. д. При каждом следующем выполнении значение i будет увеличиваться на 1. Последний раз команда **вывод** будет выполнена для i=n.

Ситуация, когда какую-то команду или группу команд нужно выполнить последовательно для всех значений в некотором диапазоне (в данном случае в диапазоне от 1 до n), встречается в информатике довольно часто, и для ее описания в алгоритмическом языке есть специальная команда — цикл **для**. Вот как с помощью этой команды можно переписать алгоритм (А58):

```
алг квадраты                                                   (А59)
нач цел n, i
| вывод "Введите количество квадратов:"; ввод n
| нц для i от 1 до n
| | вывод нс, i**2
| кц
кон
```

13.7. ОБЩИЙ ВИД ЦИКЛА для

Общий вид цикла **для** таков:

```
нц для i от i1 до i2
| тело цикла (последовательность команд)
кц
```

Здесь i1, i2 — произвольные целые числа или выражения с целыми значениями. При выполнении цикла его тело выполняется для i=i1, i=i1+1, i=i1+2, ..., i=i2. Правила алгоритмического языка допускают задание любых целых i1, i2. В частности, i2 может быть меньше i1. Этот случай не считается ошибочным — просто тело цикла не будет выполнено ни разу, а ЭВМ сразу перейдет к выполнению команд, записанных после **кц**.

13.8. ДВА ПРИМЕРА АЛГОРИТМОВ С ЦИКЛОМ для

```
алг цел сумма квадратов (арг цел n)                    (A60)
 дано n⩾1
 надо | знач = 1**2+2**2+...+n**2
нач цел i
| знач :=0
| нц для i от 1 до n
| | знач := знач +i**2
| кц
кон
```

Факториалом натурального числа n (обозначается n!) называется произведение $1\cdot2\cdot3\cdot4\cdot\ldots\cdot n$.

```
алг цел факториал (арг цел n)                          (A61)
 дано n⩾1
 надо | знач =1·2·3·4·...·n
нач
| знач :=1
| нц для k от 1 до n
| | знач := знач *k
| кц
кон
```

УПРАЖНЕНИЯ

1. Составьте алгоритм, который:

а) запрашивает у человека два числа и выводит их полусумму;

б) запрашивает у человека пять целых чисел и выводит квадраты этих чисел;

в) запрашивает у человека два натуральных числа — числитель и знаменатель дроби — и выдаёт результат сокращения этой дроби (используйте в качестве вспомогательного алгоритм упражнения 5,б из предыдущего параграфа);

г) запрашивает у человека четыре вещественных числа a, b, c, d и выводит длину отрезка от точки (a, b) до точки (c, d);

д) запрашивает у человека координаты вершин треугольника на плоскости и выводит его периметр и площадь;

е) запрашивает у человека четыре вещественных числа a, b, c, d и с помощью Чертежника рисует отрезок от точки (a, b) до точки (c, d);

ж) запрашивает у человека три вещественных числа a, b, r и с помощью Чертежника приближенно рисует окружность радиуса r с центром в точке (a, b);

з) запрашивает у человека целое число n в диапазоне от 1 до 100 и сообщает, простое оно или нет.

2. Измените алгоритм "вычисление суммы вклада" (A57) так, чтобы он циклически запрашивал у человека значения a и n, вычислял и выводил общую сумму, снова запрашивал a и n и т. д. до тех пор, пока вводимое значение $n>0$. При вводе $n\leqslant0$ выполнение алгоритма должно завершиться.

3. Определите, что будет нарисовано в результате выполнения следующей последовательности команд:

а)
```
сместиться в точку (0, 0)
опустить перо
нц для k от 1 до 3
| сместиться на вектор (   k,   0)
| сместиться на вектор (   0,  -k)
| сместиться на вектор (-k-1,   0)
| сместиться на вектор (   0, k+1)
кц
```

б)
```
сместиться в точку (0, 0)
опустить перо
нц для k от 2 до 5
| сместиться на вектор (   k,    0)
| сместиться на вектор (   0, -k-1)
| сместиться на вектор (-k-2,    0)
| сместиться на вектор (   0,  k+3)
кц
```

в)
```
сместиться в точку (1, -1)
опустить перо
нц для i от -10 до 10
| сместиться в точку ((i/10)**2, (i/10)**3)
кц
```

4. Составьте алгоритм с целочисленными аргументами m и n, который с помощью Робота закрашивает клетки, отмеченные точками (рис. 62).

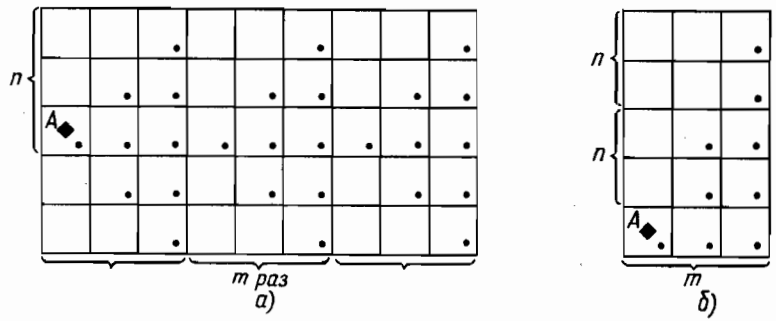


Рис. 62

5. Составьте алгоритм-функцию, который вычисляет:

а) сумму $n^2+(n+1)^2+...+(2n)^2$;

б) сумму $1!+2!+3!+...+n!$, где $k!=1\cdot2\cdot3\cdot...\cdot k$;

в) сумму $1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+...+\frac{1}{n!}$.

6. Составьте алгоритм для Чертежника, рисующий правильный n-угольник, вписанный в окружность радиуса R.

§ 14. ТАБЛИЧНЫЕ ВЕЛИЧИНЫ И РАБОТА С НИМИ

14.1. ТАБЛИЧНЫЕ ВЕЛИЧИНЫ ПОЗВОЛЯЮТ РАБОТАТЬ С БОЛЬШИМИ ОБЪЕМАМИ ИНФОРМАЦИИ

Секрет могущества ЭВМ — высокая скорость и большая память. Мы уже научились использовать скорость работы ЭВМ: команды циклов позволяют составлять короткие алгоритмы, при выполнении которых ЭВМ быстро совершает очень длинные последовательности действий. До сих пор, однако, во всех наших алгоритмах объем информации, хранимый в памяти ЭВМ в процессе выполнения алгоритма, был очень невелик — одно, два, три вещественных или целых числа. А как быть, если нужно работать с сотней, тысячей или сотней тысяч чисел? Для записи алгоритмов, работающих с большими объемами информации, в алгоритмическом языке существуют специальные ***табличные величины*** (или просто ***таблицы***).

Табличные величины состоят из других величин, как правило, целых или вещественных, называемых ***элементами***. Элементы в таблице могут быть расположены по-разному, но наиболее часто встречаются линейные таблицы.

14.2. ЛИНЕЙНЫЕ ТАБЛИЦЫ

Как и любая величина, линейная таблица занимает место в памяти ЭВМ, имеет имя и значение, например:

цел таб k

значения элементов:	9	4	0	−17	123
индексы элементов:	1	2	3	4	5

Запись **цел таб** k означает, что величина k является таблицей (**таб**) и состоит из целых (**цел**) чисел. Значение таблицы k — это пять целых чисел: 9, 4, 0, −17 и 123.

Элементы таблицы отдельных имен не имеют. Для обозначения i-го элемента таблицы k используется запись k[i].

14.3. РАБОТА С ЭЛЕМЕНТАМИ ТАБЛИЦ

цел таб k

9	4	0	−17	123
1	2	3	4	5

k [3] := k [2] + k [4]

При выполнении этой команды ЭВМ подставит вместо k [2] и k [4] значения 2-го и 4-го элементов таблицы k, т. е. числа 4 и −17, сложит их и присвоит полученное значение 3-му элементу:

цел таб k

9	4	−13	−17	123
1	2	3	4	5

k [k [2] + 1] := 0

При выполнении этой команды ЭВМ вычислит выражение k [2] + 1 = 4 + 1 = 5 и выполнит команду присваивания k [5] := 0:

цел таб k

9	4	−13	−17	0
1	2	3	4	5

14.4. ИСПОЛЬЗОВАНИЕ ТАБЛИЦ ПРИ РЕШЕНИИ ЗАДАЧ

Пусть требуется с помощью Робота исследовать радиационную обстановку в горизонтальном коридоре, например узнать число радиоактивных клеток, максимальный уровень радиации, средний уровень, число опасных клеток и т. д. Решение этой задачи удобно разбить на два этапа — сбор информации и ее обработку, т. е.:

1) провести Робота по коридору и запомнить уровни радиации во всех клетках (используя таблицу с вещественными элементами);

2) обработать запомненную в памяти ЭВМ информацию и вычислить требуемые характеристики коридора (на этом этапе Робот не нужен).

14.5. АЛГОРИТМ СБОРА ИНФОРМАЦИИ ОБ УРОВНЯХ РАДИАЦИИ

```
алг измерение радиации (арг цел n, рез вещ таб a[1:n]) (A62)
 дано |Робот стоит в левой клетке горизонтального коридора
      |длины n, уходящего вправо
 надо |Робот вышел из коридора вправо, в таблице a запомнены
      |уровни радиации всех клеток коридора
нач цел i
| нц для i от 1 до n
| | a[i]:=радиация; вправо
| кц
кон
```

Запись "**вещ таб** a[1:n]" означает, что таблица (**таб**) a состоит из n вещественных (**вещ**) элементов, занумерованных числами от 1 до n.

Напомним, что по команде "радиация" Робот сообщает ЭВМ уровень радиации в той клетке, где он находится. Поскольку после запоминания уровня радиации (команда "a[i]:=радиация") Робот переходит в следующую клетку коридора (команда "вправо"), то при выполнении алгоритма сначала в a[1] будет запомнен уровень радиации первой клетки коридора, затем в a[2] — уровень радиации второй клетки и т. д. После выхода Робота из коридора в таблице a окажется вся информация, необходимая для радиационной разведки коридора.

14.6. АНАЛИЗ ТАБЛИЧНОЙ ИНФОРМАЦИИ

Как только информация об уровнях радиации клеток в коридоре собрана в таблицу, задачи радиационной разведки превращаются в задачи анализа табличной информации. Например, подсчет числа радиоактивных клеток сводится к подсчету числа положительных элементов таблицы, определение максимального

уровня радиации — к определению максимального элемента в таблице и т. д.

Поэтому отвлечемся на время от задач радиационной разведки и рассмотрим несколько алгоритмов анализа табличной информации (A63—A65). Оформим каждый из них как алгоритм-функцию с двумя аргументами: **цел** n, **вещ таб** a[1:n].

14.7. ЧИСЛО ПОЛОЖИТЕЛЬНЫХ ЭЛЕМЕНТОВ

```
алг цел число положительных (арг цел n, вещ таб a[1:n]) (A63)
 надо | знач = число элементов таблицы, больших 0
нач цел i
| знач:=0
| нц для i от 1 до n
| | если a[i]>0
| | | то знач := знач+1
| | все
| кц
кон
```

14.8. СУММА ЭЛЕМЕНТОВ

```
алг вещ сумма (арг цел n, вещ таб a[1:n])                (A64)
 надо | знач = сумма элементов таблицы a
нач цел i
| знач:=0
| нц для i от 1 до n
| | знач := знач + a[i]
| кц
кон
```

14.9. МАКСИМУМ

```
алг вещ максимум (арг цел n, вещ таб a[1:n])             (A65)
 надо | знач = максимальное число в таблице a
нач цел i
| знач:=a[1]
| нц для i от 2 до n
| | если a[i]>знач
| | | то знач := a[i]
| | все
| кц
кон
```

В процессе выполнения этого алгоритма величина **знач** равна максимуму из уже обработанных элементов таблицы. До цикла **знач** =a[1]. Внутри цикла при обработке каждого нового элемента проводится сравнение этого элемента с максимумом из

ранее обработанных (величиной **знач**). Если a[i] > **знач**, то **знач** корректируется, т. е. заменяется на новый максимум, равный a[i]. По завершении цикла обработаны все элементы, поэтому **знач** равно максимуму среди всех элементов таблицы a.

14.10. РАДИАЦИОННАЯ РАЗВЕДКА КОРИДОРА

Вернемся теперь к задаче разведки коридора. Пусть, например, требуется разведать число радиоактивных клеток и средний уровень радиации. Для решения этой задачи можно составить следующий алгоритм:

```
алг разведка (арг цел n, рез цел m, рез вещ r)                (A66)
 дано n>0 |Робот в левой клетке горизонтального коридора
          |длины n, уходящего вправо
 надо |Робот вышел из коридора вправо, m= число клеток, в ко-
      |торых уровень радиации больше 0,r=средний уровень
      |радиации в коридоре
нач вещ таб a[1:n]
| измерение радиации (n, a)
|   утв |таблица a содержит значения уровней радиации в клет-
|       |ках коридора, Робот в клетке выхода
| m:=число положительных (n, a)
| r:=сумма (n, a)/n
кон
```

В этом алгоритме величина **вещ таб** a является промежуточной — место для нее в памяти ЭВМ отводится только на время выполнения алгоритма.

Например, при вызове "разведка (10, m1, r1)" ЭВМ запомнит значение аргумента n=10 и, встретив описание **вещ таб** a[1:n], отведет внутри прямоугольника алгоритма "разведка" часть памяти для таблицы a из 10 вещественных элементов. Аналогично, при вызове "разведка (20, m2, r2)" ЭВМ запомнит, что n=20, и по описанию **вещ таб** a[1:n] отведет место для таблицы из 20 элементов.

Алгоритм "разведка" использует вспомогательный алгоритм "измерение радиации" (A62), а также вспомогательные алгоритмы-функции "число положительных" (A63) и "сумма" (A64).

14.11. ПОИСК ЭЛЕМЕНТА В ТАБЛИЦЕ

Нахождение индекса i элемента a[i] по его значению x (т. е. нахождение такого i, что a[i]=x) принято называть ***поиском*** элемента x в таблице a. Простейший алгоритм поиска такой: будем перебирать все элементы таблицы от начала до конца (т. е. от a[1] до a[n]). Если очередной элемент равен x, то запомним его индекс в величине i:


```
алг поиск (арг цел n, вещ таб a[1:n], вещ x, рез цел i)        (A67)
 надо |a[i]=x, если значение x встречается в таблице a,
      |i=0 в противном случае
нач цел j
| i:=0
| нц для j от 1 до n
| | если a[j]=x
| | то i:=j
| | все
| кц
кон
```

Если в таблице нет элементов, равных x, то значение величины i внутри цикла не изменится и останется равным 0. Таким образом, после окончания выполнения i будет равно либо 0 (если элемента нет), либо искомому индексу элемента x.

14.12. ИНДЕКС МАКСИМАЛЬНОГО ЭЛЕМЕНТА

```
алг цел индекс максимума (арг цел n, цел таб a[1:n])          (A68)
 надо | знач = индекс максимального элемента в таблице a
нач цел i
| знач :=1
| нц для i от 2 до n
| | если a[i]>a[знач]
| | то знач :=i
| | все
| кц
кон
```

В процессе выполнения этого алгоритма величина **знач** равна индексу максимального из уже обработанных элементов. После окончания выполнения алгоритма все элементы таблицы обработаны и величина **знач** равна индексу максимального из них.

14.13. ПРЯМОУГОЛЬНЫЕ ТАБЛИЦЫ

Прямоугольные таблицы отличаются от линейных только тем, что их элементы располагаются не в одну линию, а по двум направлениям в виде прямоугольника:

вещ таб b

1	7.15	2.71828	−4.56	−17	0
2	3.14	−9.81	10.11	23.25	−91
3	5.44	161	−4.5	256	123
	1	2	3	4	5

Чтобы указать элемент прямоугольной таблицы, надо использовать не один индекс, а два: номер строки и номер столбца, на пересечении которых находится элемент. Например, в приведенной таблице b[3,2] = 161, а b[2,3] = 10.11.

При описании таблицы надо указать диапазон изменения каждого индекса. Например, для приведенной выше таблицы b описание будет таким: **вещ таб** b[1:3, 1:5]. Это означает, что таблица b содержит 3 строки и 5 столбцов.

УПРАЖНЕНИЯ

1. Известно, что значением целочисленной табличной величины k является четверка чисел (3, 1, −2, 4). Чему равно:
а) k[1]; б) k[3]; в) k[5]; г) k[k[1]]?

2. Известно, что значением целочисленной табличной величины k является четверка чисел (3, 1, −2, 4). Вычислите значение величины k после выполнения серии команд:
а) k[1] := k[4]; k[1] := k[1] + k[2]
б) i := 3; k[1] := k[i] * k[i + 1]

3. Задана линейная таблица k[1:4]. Запишите арифметическое выражение, значением которого является:
а) удвоенное значение третьего элемента таблицы;
б) сумма первого и второго элементов таблицы;
в) произведение всех элементов таблицы.

4. Как будет работать алгоритм "поиск" (А67), если элемент x встречается в таблице дважды, на первом и на последнем местах? Чему будет равно i, если элемент x встречается в таблице несколько раз?

5. Составьте алгоритмы со следующими заголовками:

а) **алг** обмен (**арг цел** n, **арг рез вещ таб** a[1:n])
надо | первый и последний элементы таблицы поменялись
|местами

б) **алг** перестановка (**арг цел** n, **арг рез вещ таб** a[1:n])
надо |минимальный элемент таблицы переставлен в ее начало,
|а первый — на место минимального

в) **алг вещ** минимум (**арг цел** n, **вещ таб** a[1:n])
надо | **знач** = значение минимального элемента в таблице a

г) **алг цел** индекс минимума (**арг цел** n, **вещ таб** a[1:n])
надо | **знач** = индекс минимального элемента в таблице a

д) **алг цел** число больших (**арг цел** n, **вещ таб** a[1:n], **вещ** x)
надо | **знач** = число элементов таблицы a, больших x

е) **алг** копия (**арг цел** n, **вещ таб** a[1:n], **рез вещ таб** b[1:n])
надо |таблица b содержит копию элементов таблицы a

ж) **алг** копия (**арг цел** n, **вещ таб** a[1:n], **рез вещ таб** b[1:n])
надо |элементы таблицы a скопированы в таблицу b
|в обратном порядке: b[1] = a[n], b[2] = a[n−1], ...

з) **алг** корректировка (**арг цел** n, **арг рез цел таб** a[1:n])
надо |все элементы отчета a, меньшие 100, заменены на 100

и) **алг** заполнение нулями (**арг цел** n, **рез цел таб** a[1:n])
надо |все элементы a равны нулю

6. Составьте алгоритмы со следующими заголовками:

а) **алг цел** произведение (**арг цел** n, **арг цел таб** a[1:n])
надо | **знач** = произведение элементов таблицы a

б) **алг вещ** среднеарифметическое (**арг цел** n, **вещ таб** a[1:n])
надо | **знач** = среднее арифметическое элементов a

в) **алг** график (**арг вещ** a, b, **цел** n, **вещ таб** y[1:n])
дано a < b **и** n ⩾ 2 |в таблице y заданы n значений функции
|на отрезке [a, b] в равноотстоящих точках
надо |Чертежник построил приближенный график функции
|на отрезке [a, b] в виде ломаной с n вершинами

г) **алг** ломаная (**арг цел** n, **вещ таб** x[1:n], y[1:n])
дано n ⩾ 2 |в таблицах x и y заданы координаты вершин
|ломаной линии с n вершинами
надо |Чертежник нарисовал эту ломаную линию

д) **алг** простые числа (**арг цел** n, **рез цел таб** p[1:n])
дано n ⩾ 1
надо |в таблице p — первые n простых чисел

е) **алг** упорядочение (**арг цел** n, **арг рез цел таб** a[1:n])
надо |элементы a упорядочены по неубыванию,
|т. е. переставлены так, что a[1] ⩽ a[2] ⩽ ... ⩽ a[n]

7. Составьте алгоритмы со следующими заголовками:

а) **алг** уровни радиации (**арг цел** m, n, **рез вещ таб** a[1:m, 1:n])
дано |Робот в левом верхнем углу прямоугольника
|размером m·n клеток, внутри стен нет
надо |в таблице a запомнены уровни радиации клеток
|прямоугольника, Робот в исходном положении

б) **алг вещ** сумма (**арг цел** m, n, **вещ таб** a[1:m, 1:n])
надо | **знач** = сумма элементов таблицы a

8. Используя алгоритмы предыдущего упражнения как вспомогательные, составьте по образцу алгоритма А66 алгоритм:

алг разведка прямоугольника (**арг цел** m, n, **рез вещ** r)
дано m > 0 **и** n > 0 |Робот в левом верхнем углу прямоугольника
|размером m·n клеток, внутри стен нет
надо |Робот в исходном положении,
|r = средний уровень радиации в прямоугольнике

§ 15. ЛОГИЧЕСКИЕ, СИМВОЛЬНЫЕ И ЛИТЕРНЫЕ ВЕЛИЧИНЫ

15.1. ТИП ВЕЛИЧИНЫ

Мы уже видели, что в алгоритмическом языке есть величины разных типов, например **цел** (значение величины — целое число) и вещ (значение величины — вещественное число). В предыдущем параграфе мы познакомились также с табличными величинами, состоящими из целых или вещественных чисел. В этом параграфе мы изучим ***логические*** (**лог**) и ***символьные*** (**сим**) величины, а также символьные таблицы, которые в алгоритмическом языке называются ***литерными*** (**лит**) величинами.

Тип величины (**цел**, **вещ**, **цел таб**, **лог** и пр.) определяет: 1) какие значения может принимать величина в процессе выполнения алгоритма; 2) в каких выражениях и как эту величину можно использовать.

15.2. ЛОГИЧЕСКИЕ ВЕЛИЧИНЫ, ВЫРАЖЕНИЯ И ПРИСВАИВАНИЯ

Логическая величина может принимать только одно из двух ***логических значений*** — либо **да**, либо **нет** (либо может быть не определена). Для установки или изменения значения величины можно использовать команду ***логического присваивания:***

имя величины := условие (логическое выражение)

По аналогии с арифметическими выражениями условия часто называют также ***логическими выражениями***. С общей формой записи условий мы уже знакомились в § 10. Например, для величины **лог** q можно написать

q := 0 ⩽ t ⩽ 1 **и** снизу стена

При выполнении команды логического присваивания ЭВМ сначала проверяет записанное справа условие (говорят также ***вычисляет логическое выражение***). Условие проверяется так же, как проверяются условия в командах **пока** и **если**. В результате проверки получается либо **да** (условие соблюдается), либо **нет** (условие не соблюдается). Полученное значение (т. е. либо **да**, либо **нет**) и записывается внутрь прямоугольника величины:

Логические величины можно использовать в логических выражениях (т. е. в условиях) наравне со сравнениями и вызовами команд-вопросов, например:

если q **то** ...
нц пока q **или** клетка закрашена ...

При проверке таких условий ЭВМ просто подставляет вместо имени логической величины ее значение.

15.3. ПРИМЕР АЛГОРИТМА С ЛОГИЧЕСКИМИ ВЕЛИЧИНАМИ

```
алг обработка тупика                                          (А69)
  дано |Робот левее горизонтального тупика (рис. 63)
  надо |Робот в исходном положении;
       |если в тупике была хоть одна закрашенная клетка, то
       |закрашен весь тупик
нач лог q
| q:=нет                  |закрашенные клетки пока не обнаружены
| нц пока справа свободно
| | вправо
| | если клетка закрашена
| | то q:=да                          |обнаружена закрашенная клетка
| | все
| кц
| утв |Робот в самой правой клетке тупика,
| |   |q=(в тупике есть закрашенная клетка)
| нц пока снизу стена |т. е. пока в тупике
| | если q |т. е. если была хоть одна закрашенная клетка
| |   |то закрасить
| | все
| | влево
| кц
кон
```

В этом алгоритме при движении Робота вправо по тупику информация о том, встретил ли он хотя бы одну закрашенную клетку, запоминается с помощью логической величины q. На обратном пути эта информация используется.

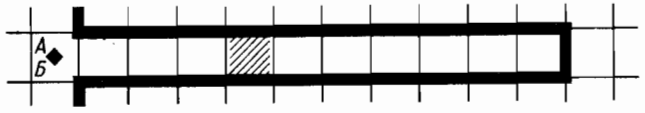


Рис. 63

15.4. ПРИМЕР ЛОГИЧЕСКОГО АЛГОРИТМА-ФУНКЦИИ

```
алг лог перекресток                                           (А70)
  надо |если из клетки можно уйти в любом направлении, то
       |знач =да. В противном случае знач = нет
```

```
нач
| знач: =справа свободно и слева свободно и сверху свободно и
|       снизу свободно
кон
```

Используя этот алгоритм-функцию как вспомогательный, можно составить, например, следующий алгоритм:

```
алг лог вправо до перекрестка                                   (А71)
 дано |Робот в горизонтальном коридоре, левее перекрестка
 надо |Робот на перекрестке
нач
| нц пока не перекресток
|  вправо
| кц
кон
```

При проверке условия "**не** перекресток" в цикле **пока** ЭВМ будет вызывать вспомогательный алгоритм-функцию А70 и подставлять вычисленное значение вместо имени функции "перекресток".

15.5. ИСПОЛЬЗОВАНИЕ ЛОГИЧЕСКОГО АЛГОРИТМА-ФУНКЦИИ В МЕТОДЕ ПОСЛЕДОВАТЕЛЬНОГО УТОЧНЕНИЯ

Пусть известно, что Робот находится в левой клетке горизонтального коридора. Из некоторых клеток коридора вверх отходят тупики. Требуется закрасить клетки коридора, от которых вверх отходят тупики длиной больше 3 клеток (рис. 64).

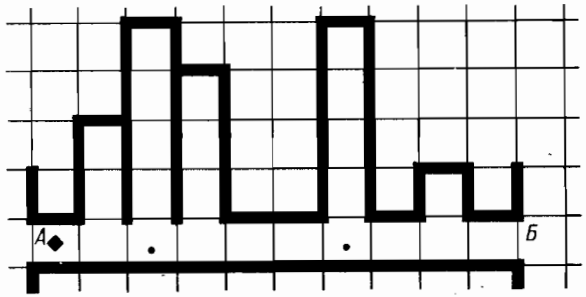


Рис. 64

Для решения этой задачи мы можем провести Робота по коридору слева направо, в каждой клетке коридора проанализировать, не выходит ли из нее длинный тупик, и если выходит — то закрасить клетку. Используя метод последовательного уточнения, этот план можно почти дословно записать так:


```
алг разметка коридора                                           (А72)
 дано |Робот в левой клетке горизонтального коридора, из неко-
      |торых клеток коридора вверх отходят тупики
 надо |Робот вышел из коридора вправо, клетки коридора на-
      |против длинных тупиков закрашены
нач
| нц пока снизу стена |т. е. пока Робот в коридоре
|  если сверху длинный тупик
|  | то закрасить
|  все
|  вправо
| кц
кон
```

Здесь использован вызов еще не написанного вспомогательного логического алгоритма-функции "сверху длинный тупик". Теперь, в соответствии с методом последовательного уточнения, составим этот алгоритм-функцию:

```
алг лог сверху длинный тупик                                    (А73)
 надо |знач=да, если вверх отходит тупик длины больше 3,
      |Робот в исходном положении
нач цел n
| n:=0
| нц пока сверху свободно
|  вверх; n:=n+1
| кц
| вниз на (n)
| знач :=(n>3)
кон
```

В результате сравнения $n>3$ ЭВМ получает либо **да**, либо **нет**. Это логическое значение присваивается переменной **знач**.

15.6. СИМВОЛЬНЫЕ ВЕЛИЧИНЫ

Значением символьной (**сим**) величины является один символ: русская или латинская большая или маленькая буква, цифра, знак препинания или специальный знак (например, "+", "−", "*", "/", "<", "=" и др.).

Существует также символ " ", который называется ***пробелом*** и используется для разделения слов в последовательности символов.

Всего символьная величина в алгоритмическом языке может принимать 256 различных значений, т. е. всего в алгоритмическом языке существует 256 различных символов. Их принято нумеровать от 0 до 255. Номер символа называется его ***кодом***.

В алгоритме ***символьные*** значения записываются в кавычках, например, "а", "+", "4". Таким образом, для величины **сим** s можно написать:

```
s:="4"
если s="+" то ...
нц пока s≠"а" ...
```

15.7. ЛИТЕРНЫЕ ВЕЛИЧИНЫ

Линейные таблицы, элементами которых являются символы, в алгоритмическом языке называются ***литерными*** (**лит**) величинами. Значением литерной величины является строка символов переменной длины. Для обозначения i-го элемента литерной величины t используется запись t[i].

В алгоритме литерные значения записываются в виде строки символов, заключенных в кавычки, например:

t:="информатика"

После этой команды t[1]="и", t[2]="н", t[3]="ф", ..., t[11]="а":

лит t

и	н	ф	о	р	м	а	т	и	к	а
1	2	3	4	5	6	7	8	9	10	11

После выполнения команды

t:="МИР"

значением литерной величины t станет последовательность символов длины 3

лит t

М	И	Р
1	2	3

15.8. ДЛИНА ЛИТЕРНОЙ ВЕЛИЧИНЫ

Число символов в строке t называется ее ***длиной***. Для обозначения длины используется запись **длин** (t). Длина может меняться в процессе выполнения алгоритма. Строка может не содержать ни одного символа. В этом случае **длин** (t)=0. Такую строку можно записать как "", т. е. ноль символов между кавычками. Строка " ", состоящая из одного пробела, имеет длину 1 и отличается от строки "".

Рассмотрим несколько примеров работы с литерными величинами.

15.9. СКОЛЬКО РАЗ В СТРОКЕ ВСТРЕЧАЕТСЯ СИМВОЛ X

```
алг цел количество символов (арг сим x, арг лит a)        (A74)
 надо | знач = количество символов x в строке a
нач цел i
| знач :=0
| нц для i от 1 до длин (a)
| | если a[i]=x
| | | то знач := знач +1
| | все
| кц
кон
```

Для того чтобы с помощью этого алгоритма узнать общее количество k восклицательных и вопросительных знаков в строке b, достаточно написать команду присваивания:

k:=количество символов ("!", b)+количество символов ("?", b)

15.10. ДОЛЯ ПРОБЕЛОВ В СТРОКЕ

Чтобы определить долю пробелов в строке, надо количество пробелов поделить на общее количество символов (длину строки). Воспользуемся для подсчета количества пробелов алгоритмом A74:

```
алг вещ доля пробелов (арг лит a)                          (A75)
 дано длин (a)⩾1
 надо | знач = доля пробелов в строке a
нач
| знач :=количество символов (" ", a)/длин (a)
кон
```

15.11. ЗАМЕНА ОДНОГО СИМВОЛА НА ДРУГОЙ

```
алг замена (арг сим x, y, арг рез лит a)                   (A76)
 надо |в строке a все символы x заменены на y
нач цел i
| нц для i от 1 до длин (a)
| | если a [i]=x
| | | то a [i]:=y
| | все
| кц
кон
```

В алгоритме (А76) величина a является одновременно аргументом и результатом (отсюда и запись **арг рез лит** a). Пусть, например,

b="Понял! Немедленно выезжаю."

Тогда после вызова

замена ("·", "!", b)

получим:

b="Понял! Немедленно выезжаю!"

А если после этого вызвать

замена ("е", "и", b)

то получим:

b="Понял! Нимидлинно выизжаю!"

15.12. ОПЕРАЦИЯ СОЕДИНЕНИЯ

Операция соединения позволяет соединить две строки в одну и обозначается знаком "+". Пусть

a="план",
b=" не",
c=" выполнен",

а x — процент выполнения плана. Тогда после команды

```
если x<100
| то t:=a+b+c
| иначе t:=a+c
все
```

при x<100 значение литерной величины t будет равно "план не выполнен", а при x⩾100 значение будет равно "план выполнен".

15.13. ВЫРЕЗКИ

До сих пор мы работали со строками либо целиком, либо поэлементно. Операция ***вырезки*** позволяет "вырезать" из строки группу соседних символов.

Вырезка из строки t обозначается t [i:j] (читается "t от i до j"), где i — индекс первого, а j — индекс последнего элементов вырезки. Например, если значением литерной величины t является строка "информатика", то

t [1:4] ="инфо"
t [8:10] ="тик"
t [10:10] +t [1:2] +t [4] ="кино"
t [1:4] +"МИР"="инфоМИР"
t [10:10] ="k"

15.14. КОМАНДА ПРИСВАИВАНИЯ ВЫРЕЗКЕ

Вырезку можно использовать в левой части команды присваивания. Например, если t="План недовыполнен", то при выполнении команды присваивания

t [6:9] :="пере"

элементы строки t с шестого по девятый будут заменены на "пере" и t станет равным "План перевыполнен".

15.15. ПРИМЕР АЛГОРИТМА, ИСПОЛЬЗУЮЩЕГО ВЫРЕЗКИ

(А77)

```
алг замена (арг рез лит t)
 надо |в строке t "1990" всюду заменено на "2000"
нач цел i
| нц для i от 1 до длин (t) −3
| | если t [i:i+3] ="1990"
| | то t [i:i+3] :="2000"
| | все
| кц
кон
```

УПРАЖНЕНИЯ

1. Определите значение величины q после выполнения каждой из следующих команд присваивания:

а) q:=**да**
б) q:=**не** q
в) q:=a<b<c **и** q
г) q:=q **или** (a+b=c)
д) q:=0>1
е) q:=1>0

если перед выполнением a=1, b=2, c=3, q=**нет**.

2. Составьте алгоритмы со следующими заголовками:

а) **алг лог** клетка опасна (**арг вещ** a)
дано |на поле Робота стен нет
надо | **знач** = **да**, если суммарный уровень радиации в клетке с Роботом и в ее 8 соседях превышает a;
|положение Робота не изменилось

б) **алг лог** четно (**арг цел** n)
надо | **знач**=**да**, если n четно

3. Что произойдет при попытке выполнить алгоритм А72 в обстановке, когда от коридора вверх отходит другой коридор (не тупик)?

4. Измените алгоритм А73 так, чтобы при выполнении алгоритма А72 закрашивались клетки напротив любых ответвлений вверх (не обязательно тупиков) длиною больше 3 клеток.

5. Составьте алгоритм со следующим заголовком:

алг число проходов (**рез цел** n)
дано | Робот в левом нижнем углу прямоугольника, огороженного
| стенами. В нижней стене прямоугольника есть несколько
| проходов разной ширины
надо | Робот в правом нижнем углу прямоугольника,
| n = число проходов в нижней стене

6. Строка t равна "апельсин". Чему равны строки
а) t[2:3] + t[8] + t[5];
б) t [6] + t [2] + t [1] + t [8] + t [7] + t [3:5] ?

7. Строка t равна "вертикаль". По образцу упражнения 6 составьте из t слова а) "ветка"; б) "кирка"; в) "кильватер".

8. Составьте алгоритмы со следующими заголовками:

а) **алг цел** количество предложений (**арг лит** x)
дано | строка x состоит из нескольких предложений, каждое
| из которых кончается точкой, восклицательным или
| вопросительным знаком
надо | **знач** = количество предложений в строке x

б) **алг** обмен (**арг сим** x, y, **арг рез лит** a)
надо | всюду в "a" x заменен на y, а y заменен на x

в) **алг лит** перевертыш (**арг лит** a)
надо | **знач** = "перевертыш", если строка a совпадает с
| собой после переворачивания, и **знач** = "не перевертыш"
| в противном случае

г) **алг** замена (**арг лит** x, y, **арг рез лит** a)
дано **длин** (x) = **длин** (y)
надо | всюду в строке a x заменено на y

д) **алг** переворачивание (**арг лит** a, **рез лит** b)
надо | строку a записать в обратном порядке в строку b

9. Составьте алгоритмы, которые по строке a получают зашифрованную строку b, по следующим правилам:
а) каждая буква от "а" до "ю" заменяется на следующую по алфавиту, каждая буква "я" заменяется на букву "а";
б) первая буква алфавита заменяется на одиннадцатую, вторая — на двенадцатую, ..., последняя — на десятую;
в) после каждой согласной вставляется буква "а";
г) после каждой согласной вставляется слог "ла".

10. Составьте алгоритмы, которые по зашифрованной строке b получают расшифрованную строку a (см. упражнение 9).

11. Придумайте способ шифровки, при котором каждая пара букв заменяется либо на одну, либо на три буквы. Составьте алгоритмы шифровки и дешифровки для этого способа.

§ 16. СОСТАВЛЕНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Программирование циклов, т. е. запись длинной последовательности действий в виде короткого алгоритма,— сложная задача. В этом параграфе мы рассмотрим несколько простейших методов составления циклических алгоритмов: рекуррентные соотношения, однопроходные алгоритмы, инвариант и рекурсию.

16.1. РЕКУРРЕНТНЫЕ СООТНОШЕНИЯ

И в математике и в информатике часто встречаются последовательности чисел, в которых каждый следующий член выражается через предыдущие.

В арифметической прогрессии, например, каждый следующий член равен предыдущему, увеличенному на разность прогрессии:

$$a_i = a_{i-1} + d.$$

В последовательности 1, 1, 2, 3, 5, 8, 13, ... (она называется ***последовательностью Фибоначчи***) каждый следующий член равен сумме двух предыдущих. Для этой последовательности

$$a_i = a_{i-1} + a_{i-2}, \qquad a_1 = a_2 = 1.$$

Формулы, выражающие очередной член последовательности через один или несколько предыдущих членов, называются ***рекуррентными соотношениями***.

Как вычислить n-й член последовательности, заданной рекуррентным соотношением? Иногда для n-го члена есть простая формула. Например, для арифметической прогрессии $a_n = a_1 + d*(n-1)$. Чаще, однако, такой простой формулы нет или она неизвестна. В этом случае члены последовательности вычисляют по рекуррентному соотношению один за другим от i = 1 до i = n.

16.2. РЕКУРРЕНТНЫЕ ВЫЧИСЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ ТАБЛИЦ

Рассмотрим последовательность 2, 3, 5, 9, 17, ..., задаваемую рекуррентным соотношением

$$a_i = 2*a_{i-1} - 1, \qquad a_1 = 2.$$

Составим простейший алгоритм для вычисления n-го члена последовательности. Будем вычислять члены последовательности по рекуррентному соотношению один за другим от i = 1 до i = n, запоминая значение i-го члена последовательности в элементе a [i] целочисленной таблицы a:

```
алг цел элемент (арг цел n)                                    (A78)
 дано n>0
 надо | знач=n-й элемент последовательности
нач цел i, цел таб a[1:n]
| a[1]:=2
| нц для i от 2 до n
| | a[i] :=2*a[i-1]-1
| кц
| знач :=a[n]
кон
```

Хотя нас интересует лишь n-й член последовательности, в процессе выполнения алгоритма A78 ЭВМ вычисляет и хранит в памяти все члены последовательности от 1-го до n-го.

16.3. РЕКУРРЕНТНЫЕ ВЫЧИСЛЕНИЯ БЕЗ ИСПОЛЬЗОВАНИЯ ТАБЛИЦ И «ИСЧЕЗНОВЕНИЕ» ИНДЕКСОВ

Заметим, что в рассмотренном примере для вычисления следующего члена последовательности нужно знать только значение предыдущего. Поэтому можно не запоминать все члены последовательности в таблице, а иметь одну целочисленную величину a и при увеличении i изменять ее значение: a:=2*a-1.

Другими словами, рекуррентное соотношение $a_i=2*a_{i-1}-1$ можно заменить соотношением $a_{новое}=2*a_{старое}-1$, которое записывается в виде команды присваивания a:=2*a-1:

```
алг цел элемент (арг цел n)                                    (A79)
 дано n>0
 надо | знач = n-й элемент последовательности
нач цел i, a
| a:=2          |запоминание 1-го члена последовательности
| нц для i от 2 до n
| | a:=2*a-1    |вычисление i-го члена последовательности
| кц
| знач :=a
кон
```

При таком подходе используемые в рекуррентных соотношениях индексы обозначают разные значения одной и той же величины a и ***исчезают*** при записи алгоритма на алгоритмическом языке.

16.4. МЕТОД РЕКУРРЕНТНЫХ СООТНОШЕНИЙ

В общем случае, при вычислении значения какой-нибудь величины z, можно попытаться представить z в виде элемента некоторой последовательности, заданной рекуррентным соотношением. Если это удастся, то z можно вычислить рекуррентно, составив алгоритм, аналогичный A79.

Пусть, например, требуется определить общее сопротивление гирлянды из n параллельно соединенных одинаковых лампочек (каждая сопротивлением R2), если соединительные провода имеют сопротивление R1 каждый (рис. 65):

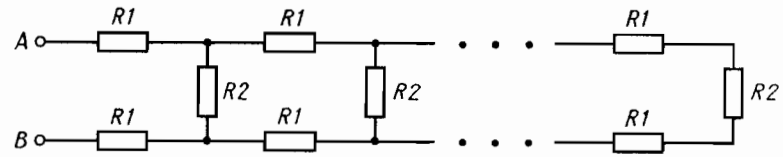


Рис. 65

Применим метод рекуррентных соотношений. Обозначим сопротивление гирлянды с i лампочками через a_i.

Рассмотрим сначала схему при i=1 (рис. 66, *а*). Очевидно, что в этом случае $a_1=2\cdot R1+R2$.

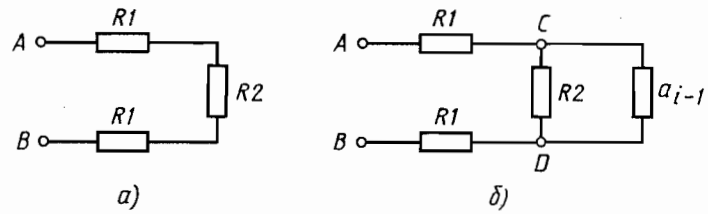


Рис. 66

Выразим теперь a_i через a_{i-1}. Схема для a_i эквивалентна схеме, изображенной на рисунке 66, *б*. Сопротивление такой схемы уже нетрудно определить: участки AC, CD и DB соединены последовательно, на участке CD параллельно соединены сопротивления R2 и a_{i-1}. Поэтому

$$a_i=2\cdot R1+\frac{1}{\frac{1}{R2}+\frac{1}{a_{i-1}}}=2\cdot R1+\frac{a_{i-1}\cdot R2}{a_{i-1}+R2}.$$

Таким образом, мы получили рекуррентное соотношение и можем записать алгоритм для вычисления сопротивления исходной схемы в виде:

```
алг вещ сопротивление (арг вещ R1, R2, арг цел n)          (A80)
 дано n>0 и R1>0 и R2>0
 надо | знач=сопротивление схемы
нач вещ a, цел i
| a:=2*R1+R2
| нц для i от 2 до n
| | a:=2*R1+(a*R2)/(a+R2)
| кц
| знач:=a
кон
```

16.5. РЕКУРРЕНТНЫЕ ВЫЧИСЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ПРОМЕЖУТОЧНЫХ ВЕЛИЧИН

Вспомним рекуррентное соотношение, задающее последовательность Фибоначчи:

$$a_i = a_{i-1} + a_{i-2}, \qquad a_1 = a_2 = 1.$$

Очередной элемент здесь выражается не через один, а через два предыдущих элемента. Поэтому в процессе вычисления надо хранить значения двух элементов.

Будем это делать с помощью двух целочисленных величин: a и b. В величине a будем хранить значение a_{i-1}, в величине b — значение a_{i-2}.

При увеличении i значения величин меняются следующим образом:

$$\begin{matrix} a_{\text{новое}} = a_i = a_{i-1} + a_{i-2} = a_{\text{старое}} + b_{\text{старое}} \\ b_{\text{новое}} = a_{i-1} = a_{\text{старое}} \end{matrix}$$

Первое соотношение записывается в виде команды присваивания:

$$a := a + b.$$

Однако старое значение a при этом будет потеряно. Поэтому перед командой $a := a + b$ старое значение a надо запомнить в какой-нибудь другой величине, например с помощью команды $r := a$. Если это сделать, то второе соотношение можно записать в виде

$$b := r.$$


```
алг цел Фибоначчи (арг цел n)                              (A81)
 дано n>0
 надо | знач=n-й член последовательности Фибоначчи
нач цел a, b, i, r
| если n⩽2 то знач:=1 иначе
| | a:=1; b:=1
| | нц для i от 3 до n
| | | утв |-1=i-1 элемент; b=i-2 элемент
| | | r:=a                  | запоминаем старое значение a
| | | a:=a+b
| | | b:=r
| | | утв | a=i-й элемент; b=i-1 элемент
| | кц
| | знач:=a                 | n-й элемент
| все
кон
```

16.6. ПРОДОЛЖЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ «ВЛЕВО»

В приведенном алгоритме (A81) начальные элементы последовательности Фибоначчи (при $n \leqslant 2$) вычисляются отдельно — не так, как все остальные. В таких случаях для упрощения алгоритма можно попытаться приписать к последовательности слева несколько новых элементов так, чтобы рекуррентное соотношение оказалось применимым начиная с i=1.

Для последовательности Фибоначчи рекуррентное соотношение можно переписать в виде:

$a_{i-2} = a_i - a_{i-1}$, откуда $a_0 = a_2 - a_1 = 0$, $a_{-1} = a_1 - a_0 = 1$.

После такого продолжения "влево" соотношение

$$a_i = a_{i-1} + a_{i-2}, \qquad a_{-1} = 1, \qquad a_0 = 0$$

станет применимым начиная с i=1 и алгоритм запишется так:

```
алг цел Фибоначчи (арг цел n)                              (A82)
 дано n>0
 надо | знач=n-й член последовательности Фибоначчи
нач цел a, b, i, r
| a:=0; b:=1
| нц для i от 1 до n
| | утв | a=i-1 элемент; b=i-2 элемент
| | r:=a          | запоминаем старое значение a
| | a:=a+b
| | b:=r
| | утв | a=i-й элемент; b=i-1 элемент
| кц
| знач:=a         | n-й элемент
кон
```

16.7. ОДНОПРОХОДНЫЕ АЛГОРИТМЫ

Пусть мы участвуем в соревнованиях по рыбной ловле и побеждает тот, кто поймает самую большую рыбу. Тогда можно вести себя так: поймав первую рыбу, поместим ее в ведро. Поймав следующую, сравним ее с той, которая у нас в ведре. Если новая рыба больше, то старую выпустим в реку, а новую поместим в ведро. Если же новая рыба меньше, то мы ее отпустим, а старую оставим в ведре. Ясно, что при этом в ведре всегда будет самая большая из всех пойманных нами рыб.

Если обозначить вес рыбы в ведре v, а вес пойманной рыбы x, то выбор рыбы можно записать так:

$$v_{\text{новое}} = \max(v_{\text{старое}}, x) \quad \text{или} \quad v := \max(v, x)$$

В общем случае изложенный подход состоит в том, что при обработке очередной порции информации новые значения величин (в примере выше — величины v) выражаются через старые значения и новую информацию (x). При этом совсем неважно, откуда поступает информация. Она может быть задана линейной таблицей, получаться в результате вычислений, поступать от Робота, вводиться человеком с клавиатуры и пр.

Нахождение максимального элемента в таблице, например, можно записать так:

```
алг вещ максимум (арг цел n, вещ таб a [1:n])                (A83)
 надо | знач = максимальное число в таблице a
нач вещ v; цел i
| v:=a [1]                    |"выловили" первый элемент
| нц для i от 2 до n
| | v:=max (v, a[i])          |выбор большего из старого максимума v
| кц                          |                и нового элемента a[i]
| знач:=v
кон
```

А алгоритм определения максимального уровня радиации в коридоре можно записать так:

```
алг максимальная радиация (рез вещ v)                         (A84)
 дано | Робот в левой клетке горизонтального коридора
 надо | Робот вышел из коридора вправо (см., например, рис. 54),
      | v = максимальный уровень радиации в коридоре
нач
| v:=радиация; вправо        | обработка первой клетки
| нц пока снизу стена        | т. е. пока Робот в коридоре
| | v:=max (v, радиация)     | учет радиации в новой клетке
| | вправо                   | переход в следующую клетку
| кц
кон
```


В приведенных алгоритмах информация обрабатывается последовательно — порция за порцией, причем каждая порция (каждый элемент таблицы, каждая клетка коридора) обрабатывается ***ровно один раз.*** Такие алгоритмы называются ***однопроходными.***

16.8. ОДНОПРОХОДНЫЙ АЛГОРИТМ ПОДСЧЕТА ЧИСЛА МАКСИМУМОВ

Пусть требуется найти число максимальных элементов в таблице **вещ таб** a [1:n]. Простейшее решение такое — сначала найти максимум, а потом просмотреть элементы таблицы еще раз и подсчитать, сколько из них совпадает с найденным максимумом. При этом, однако, каждый элемент таблицы будет обработан дважды (будет сделано два прохода по таблице).

Попробуем составить однопроходный алгоритм. Для этого надо понять, какую информацию следует хранить «в ведре» и как ее менять при обработке очередного элемента.

Пусть какая-то часть таблицы уже обработана, k — количество максимумов в этой части и мы приступаем к обработке очередного элемента x. Как меняется k? Если x меньше максимума среди уже обработанных элементов, то k не меняется. Если x совпадает с максимумом, то k увеличивается на 1. Если же x больше, то он становится единственным новым максимумом, т. е. k становится равным 1.

Таким образом, при обработке очередного элемента x нам, кроме k, нужно знать еще максимум среди обработанных элементов (обозначим его m), т. е. в процессе вычислений «в ведре» нужно хранить две величины — m и k:

```
алг цел число максимумов (арг цел n, вещ таб a [1:n])        (A85)
 дано n>0
 надо | знач = число максимальных элементов в таблице a
нач вещ m, цел i, k
| m:=a [1]; k:=1              | "выловили" первый элемент
| нц для i от 2 до n
| | выбор
| | | при a [i] <m: | ничего делать не надо
| | | при a [i] =m: k:=k+1
| | | при a[i] >m: m:=a[i]; k:=1
| | все
| кц
| знач:=k
кон
```

16.9. ОДНОПРОХОДНЫЙ АЛГОРИТМ ПОДСЧЕТА КОЛИЧЕСТВА СЛОВ В СТРОКЕ

Пусть надо определить число слов в строке символов, состоящей из букв и пробелов. (Слово — это группа подряд идущих букв. Слова разделяются одним или несколькими пробелами.)

Попробуем составить однопроходный алгоритм, т. е. будем обрабатывать строку последовательно символ за символом. Какую же информацию об обработанной части строки нам придется хранить "в ведре" и как эту информацию надо менять при обработке очередного символа?

Во-первых, нужно знать количество слов. Используем для этого целочисленную величину n:

```
цел n | количество слов в обработанной части строки
```

При обработке очередного символа x он как бы дописывается справа к обработанной части строки. Количество слов при этом либо увеличивается на 1 (если x — начало нового слова), либо остается прежним (если x — продолжение ранее начатого слова или пробел). То есть, кроме количества слов n, надо знать, продолжает x ранее начатое слово или начинает новое:

```
лог q | q=да, если новая буква начнет новое слово
      | q=нет, если новая буква продолжит уже начатое слово
```

Таким образом, «в ведре» надо хранить значения величин n и q. Перед началом обработки n=0, q=да (слов пока нет, новая буква начнет слово), а алгоритм записывается так:

```
алг цел число слов (арг лит t)                                  (A86)
 надо | знач=число слов в строке t
нач цел n, i, лог q, сим x
| n:=0; q:=да
| нц для i от 1 до длин (t)
| | x:=t [i]          |"выловили" новый символ x
| | если x≠" " и q, то n:=n+1 все
| | q:=(x=" ")        |слово может начаться только после пробела
| кц
| знач:=n
кон
```

16.10. ИНВАРИАНТ ЦИКЛА

При выяснении того, какие величины и как будут меняться в цикле, удобно описать взаимосвязь между значениями величин в виде неизменного условия, которое должно быть выполнено после любого числа выполнений тела цикла. Например, в алгоритме A43 взаимосвязь между положением Робота и значением величины n можно описать условием "n=число сделанных Роботом шагов вправо". В алгоритме "индекс максимума" (A68) условие "знач=индекс максимального среди элементов a [1], a [2], ..., a [i]" описывает взаимосвязь между значениями величин знач и i. В приведенном выше алгоритме "Фибоначчи" (A82) утверждение "a=i-й элемент, b=i−1 элемент" описывает взаимосвязь между величинами a, b, i.

Такое неизменное условие, которое соблюдается после каждого выполнения тела цикла, в информатике называется ***инвариантом*** (от лат. invariant — неизменный). По окончании цикла инвариант выполняется совершенно независимо от того, сколько раз выполнялось тело цикла и что происходило при каждом выполнении. Поэтому с помощью инварианта легче понять цель выполнения цикла или составить цикл для достижения требуемой цели. Рассмотрим следующий пример:

```
алг вещ степень (арг вещ a, арг цел n)                          (A87)
 дано n≥0
 надо | знач=a ** n
нач цел k, вещ t, b
| b:=1; t:=a; k:=n
| нц пока k≠0
| | если mod (k, 2)=0
| | | то      t:=t * t; k:=k/2
| | | иначе   b:=b * t; k:=k−1
| | все
| кц
| знач:=b
кон
```

Не правда ли, алгоритм выглядит загадочно? Зачем-то введены величины k и t, почему-то проверяется, четно ли k, величина t возводится в квадрат и т. п. Непонятно, какое это все имеет отношение к исходной задаче возведения числа a в степень n.

Тем не менее этот алгоритм возводит a в степень n и к тому же очень быстро (для n=1000 цикл выполняется всего лишь 15 раз!). Почему это происходит? Ответ на этот вопрос заключен в инварианте цикла

$$b \cdot t^k = a^n,$$

который описывает взаимосвязь между значениями величин b, t и k в процессе выполнения алгоритма.

Перед выполнением цикла b=1, t=a, k=n и инвариант превращается в очевидное тождество

$$1 \cdot a^n = a^n.$$

Внутри цикла написана команда **если**, т. е. возможны два случая:

1) k четно. В этом случае выполняются команды "t:=t * t; k:=k/2". Поскольку $(t^2)^{k/2}=t^k$, то t^k не изменится и инвариант останется выполненным.

2) k нечетно. В этом случае выполняются команды "b:=b * t; k:=k−1". Поскольку $(bt)\cdot t^{k-1}=b\cdot t^k$, то $b\cdot t^k$ не изменится и инвариант также останется выполненным.

Итак, что бы ни происходило при выполнении тела цикла, инвариант сохранится. В обоих случаях k уменьшается, следовательно, цикл рано или поздно завершится. После окончания цикла инвариант будет по-прежнему выполнен, а k будет равно 0. Поэтому

$b \cdot t^0 = a^n$, т. е. b будет равно a^n, что и требовалось.

Представить a^n в виде произведения $b \cdot t^k$ можно разными способами. Одно из таких представлений является целью выполнения алгоритма ($b = a^n$, $k = 0$), а другое легко получить в начале работы ($b = 1$, $t = a$, $k = n$). При каждом повторении тела цикла мы понемногу "перекачиваем информацию" из k в b (т. е. уменьшаем k и увеличиваем b) до тех пор, пока не получим искомое конечное представление.

16.1*. РЕКУРСИЯ

В общем случае ***рекурсией*** называется ситуация, когда какой-то алгоритм прямо или через другие алгоритмы вызывает себя в качестве вспомогательного. Сам алгоритм при этом называется ***рекурсивным***.

Рассмотрим еще раз рекуррентное соотношение, задающее последовательность Фибоначчи:

$$a_i = a_{i-1} + a_{i-2}, \qquad a_1 = a_2 = 1.$$

Вычисление n-го элемента этой последовательности можно записать в виде следующего алгоритма:

```
алг цел Ф (арг цел n)                                   (A88)
 дано n>0
 надо | знач = n-й член последовательности Фибоначчи
нач
|  выбор
|  | при n=1: знач:=1
|  | при n=2: знач:=1
|  | иначе    знач:=Ф (n-1)+Ф (n-2)
|  все
кон
```

В отличие от всех написанных нами ранее алгоритмов здесь в качестве вспомогательного вызывается сам алгоритм Ф (с другими аргументами). Таким образом, алгоритм Ф является рекурсивным.

Чтобы понять, как он будет выполняться, вспомним, что на время выполнения вспомогательного алгоритма основной алгоритм приостанавливается. Таким образом, при вызовах алгоритмов в памяти ЭВМ "накапливаются" прямоугольники приостановленных алгоритмов. При рекурсивных вызовах одного и того же алгоритма Ф ЭВМ работает точно так же, т. е. при каждом вызове Ф отводит ему в памяти новый прямоугольник. По завершении алгоритма его прямоугольник стирается из памяти и ЭВМ возвращается к предыдущему прямоугольнику алгоритма Ф.

Алгоритм Ф (A88) написан прямо по рекуррентному соотношению и выглядит очень просто. Однако работает этот алгоритм весьма неэкономно. Ф (17), например, вычисляется по этому алгоритму как Ф (16) + Ф (15). Ф (16) в свою очередь вычисляется как Ф (15) + Ф (14). Таким образом, Ф (15) в процессе вычисления Ф (17) будет вычисляться два раза (Ф (14) — три, Ф (13) — пять, Ф (12) — восемь раз и т. д.). Всего при вычислении Ф (17) ЭВМ выполнит свыше тысячи, при вычислении Ф (31) — свыше миллиона, а при вычислении Ф (45) — свыше миллиарда операций сложения! Для сравнения заметим, что при вычислении 45-го члена последовательности Фибоначчи по алгоритму A82 выполняется всего лишь 45 операций сложения.

Поэтому, хотя составить рекурсивный алгоритм обычно проще нерекурсивного, пользоваться рекурсией надо осторожно.

Рекурсивные алгоритмы естественно возникают в тех случаях, когда исходную задачу удается свести к точно такой же задаче, но с другими аргументами или в другой обстановке. При этом в ряде случаев составление нерекурсивного алгоритма может оказаться крайне сложным, почти невозможным. Вот один такой пример:

```
алг закрасить область                                   (A89)
 дано | на поле Робота стен нет; число клеток, в которые Ро-
      | бот может попасть из исходного положения, двигаясь
      | только по незакрашенным клеткам, конечно
 надо | закрашены все клетки, в которые Робот мог попасть из
      | исходного положения по незакрашенным клеткам; Робот
      | в исходном положении
нач
|  если клетка не закрашена то
|  |  закрасить
|  |  вверх;  закрасить область; вниз
|  |  вправо; закрасить область; влево
|  |  вниз;   закрасить область; вверх
|  |  влево;  закрасить область; вправо
|  все
кон
```

Если исходная клетка не закрашена, то этот алгоритм четыре раза вызывает себя в качестве вспомогательного. Каждый вызов происходит в новой обстановке: исходная клетка уже закрашена, а Робот находится в новом положении.

УПРАЖНЕНИЯ

1. Составьте алгоритм вычисления суммы

$$1+\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}.$$

Указание. Обозначьте $a_i=\frac{x^i}{i!}$ и используйте соотношения

$$a_i=a_{i-1}\cdot\frac{x}{i}, \qquad a_0=1,$$
$$s_i=s_{i-1}+a_i, \qquad s_0=1.$$

В упражнениях 2—5 выпишите рекуррентные соотношения и составьте алгоритмы вычисления значений следующих величин:

2. Сумма $\frac{x^1}{1!}-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots-(-1)^n\frac{x^n}{n!}$ (при увеличении n эта сумма приближается к значению sin (x)).

3. Сопротивление между клеммами A и B в схемах, изображенных на рисунке 67, где каждая схема содержит n сопротивлений R1.

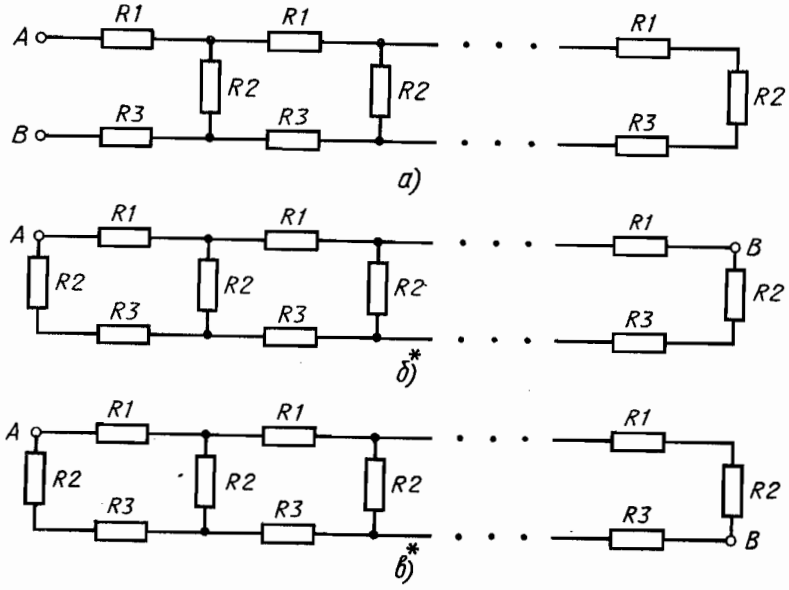


Рис. 67

Указание. Задачи б*) и в*) проще решать одновременно, с помощью одного алгоритма.

4. Электродвижущая сила и внутреннее сопротивление между клеммами A и B в схеме, состоящей из 2n соединительных проводов с сопротивлением R каждый, n одинаковых источников тока с внутренним сопротивлением r и электродвижущей силой e (рис. 68):

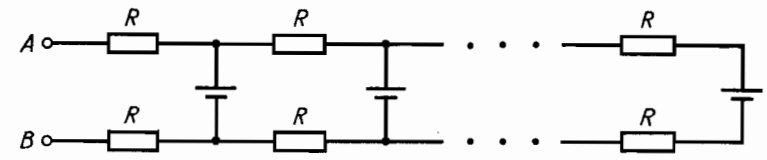


Рис. 68

5. N-й член последовательности, начинающейся с единицы, в которой:

а) каждый следующий член равен сумме обратных величин всех предыдущих;

б) каждый следующий член равен сумме квадратов всех предыдущих;

в) каждый следующий член равен синусу суммы всех предыдущих.

6. При положительном a решением уравнения $x=x/2+a/(2x)$ служит $x=\sqrt{a}$. Рекуррентное соотношение

$$x_i=\frac{x_{i-1}}{2}+\frac{a}{2x_{i-1}}, \qquad x_1=1$$

можно использовать для быстрого вычисления $\sqrt{a}$, так как элементы последовательности при увеличении i очень быстро приближаются к $\sqrt{a}$.

Вот, например, как выглядит начало этой последовательности при $a=2$ (числа изображены с 10 знаками в дробной части; начиная с x_5 эти первые 10 знаков x_i уже не меняются):

$x_1=1$
$x_2=1.5$
$x_3=1.4166666667$
$x_4=1.4142156863$
$x_5=1.4142135624$
$x_6=1.4142135624$
. . .

Составьте алгоритм:

алг вещ корень квадратный (**арг вещ** a, **арг цел** n)
дано $a\geqslant 0$ **и** $n>0$
надо | вычислен n-й член указанной выше последовательности

7. По образцу упражнения 6 составьте алгоритм для вычисления $\sqrt[3]{a}$, используя соотношение

$$x_i=\frac{2x_{i-1}}{3}+\frac{a}{3x_{i-1}^2}, \qquad x_1=1.$$

8. Составьте алгоритм, вычисляющий n-й член последовательности, заданной соотношениями:

а) $a_i=a_{i-1}+a_{i-2}+a_{i-3}$, $a_1=a_2=a_3=1$;
б) $a_i=i*a_{i-1}+a_{i-2}$, $a_1=a_2=1$;
в) $a_i=a_{i-1}+a_{i-2}+...+a_{i-k}$, $a_1=a_2=...=a_k=1$.

9. Продолжите влево последовательность 8а) на 3 элемента и 8б) на 2 элемента. Для последовательности 8в) составьте алгоритм вычисления a_{-n}, где $n>0$.

10. Выпишите рекуррентное соотношение и продолжите влево последовательности:

а) 1, 2, 6, 24, 120, 720, 5040, ...;
б) 1, 4, 11, 26, 57, 120, 247, ...;
в) 1, 1, 0, 1, −1, 2, −3, 5, −8, 13, −21, ...

11. Элементы последовательностей $x_1, x_2, ...$ и $y_1, y_2, ...$ заданы рекуррентными соотношениями:

$$x_i=x_{i-1}+0.1y_{i-1},\ x_1=1$$
$$y_i=y_{i-1}-0.1x_{i-1},\ y_1=0$$

Составьте алгоритм, который с помощью Чертежника рисует ломаную, состоящую из n звеньев, с вершинами в точках

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), ..., (x_{n+1}, y_{n+1}).$$

12. Последовательность чисел задана соотношениями

$$a_1=1,\ a_{2n}=n,\ a_{2n+1}=a_n+a_{n+1}.$$

Составьте алгоритм вычисления n-го члена этой последовательности.

13. а) В последовательности чисел пять максимальных элементов. В конец последовательности дописали число 10. Можно ли узнать, сколько максимальных элементов в новой последовательности?

б) В последовательности чисел максимальный элемент равен 7 и таких элементов 5. В конец последовательности дописали число x. Сколько максимальных элементов в новой последовательности при x=10? При x=7? При x=0?

в) Робот в левой клетке горизонтального коридора неизвестной длины. Составьте однопроходный алгоритм, который находит число клеток коридора с максимальной температурой.

14. Робот находится перед входом в тупик, идущий вправо. Составьте алгоритм, после выполнения которого положение Робота не изменится и

а) будут закрашены все клетки, уровень радиации в которых составляет более 80% от максимального;

б) будут закрашены все клетки, температура в которых выше средней.

15. а) Робот левее горизонтального коридора неизвестной длины. Известно, что клеток с нулевой температурой в коридоре нет. Составьте однопроходный алгоритм, который находит число перемен знака температуры в коридоре.

б) Функция y=f (x) не обращается в нуль в точках 1, 2, ..., n. Составьте однопроходный алгоритм, который находит число перемен знака в последовательности f (1), f (2), ..., f (n).

в) В таблице **цел таб** a [1:n] нет нулевых элементов. Составьте однопроходный алгоритм, который находит число перемен знака среди элементов таблицы.

16. Элемент таблицы называется локальным максимумом, если у него нет соседа большего, чем он сам. Составьте однопроходный алгоритм, который находит число локальных максимумов в таблице **цел таб** a [1:n].

17. Дана функция y=f (x). Составьте однопроходный алгоритм, который вычисляет, сколько раз в последовательности f (1), f (2), ..., f (n) встречается подпоследовательность а) 1, 2, 3, 4; б) 1, 2, 1, 3.

Замечание. При $n\leqslant 3$ алгоритм должен давать ответ 0. Желательно, чтобы этот случай не рассматривался особо.

18. Выпишите инвариант, описывающий взаимосвязь между значениями величин n и i при выполнении цикла

```
n:=0; i:=0
нц пока i<1000
| n:=n+2*i+1
| i:=i+1
кц
```

Чему будет равно n после завершения цикла?

19. Выпишите инвариант, описывающий взаимосвязь между значениями величин a, b, c, d при выполнении цикла

```
утв a≥0 и b>0
c:=0; d:=a
нц пока d≥b
| c:=c+1; d:=d−b
кц
```

Чему будут равны c и d после завершения цикла?

20. Известно, что в каждой из таблиц **цел таб** a [1:m], b [1:n] элементы расположены по возрастанию. Пусть x — количество элементов, встречающихся в обеих таблицах. Докажите, что равенство

x = k + число общих элементов среди a [i:m] и b [j:n]

справедливо перед выполнением и после любого числа повторений тела цикла

```
i:=1; j:=1; k:=0
нц пока i≤m и  j≤n
| выбор
|   при a [i] <b [j] : i:=i+1
|   при a [i] >b [j] : j:=j+1
|   при a [i] =b [j] : k:=k+1; i:=i+1; j:=j+1
| все
кц
```

Чему будет равно k после завершения этого цикла?

21. Докажите, что если a>0 и b>0 — целые, то после выполнения приведенных ниже команд величина c станет равной наименьшему общему кратному начальных значений величин a и b.

```
p:=b; q:=a
нц пока a≠b
| если a<b
|   то     b:=b-a; p:=p+q
|   иначе  a:=a-b; q:=q+p
| все
кц
c:=(p+q)/2
```

22. Докажите, что если a>0 и b>0 — целые, то после выполнения приведенных ниже команд величина c станет равной наибольшему общему делителю начальных значений величин a и b.

```
m:=1
нц пока a≠b
| выбор
|   при mod (a, 2) =0 и mod (b, 2) =0 : a:=div (a, 2)
|                                       b:=div (b, 2);
|                                       m:=m * 2
|   при mod (a,2) =0 и mod (b,2) ≠0          : a:=div (a,2)
|   при mod (a,2) ≠0 и mod (b,2) =0          : b:=div (b,2)
|   при mod (a,2) ≠0 и mod (b,2) ≠0 и a>b: a:=a-b
|   при mod (a,2) ≠0 и mod (b,2) ≠0 и a<b : b:=b-a
| все
кц
c:=a*m
```


23. Выпишите инвариант цикла и опишите результат выполнения команд для таблицы **вещ таб** a [1:n] :

```
i:=1; j:=n
нц пока i<j
| выбор
|   при a [i] ≥0 : i:=i+1
|   при a [j] <0 : j:=j-1
|     иначе      :t:=a[i]; a[i]:=a[j]; a[j]:=t;
|                 i:=i+1; j:=j-1
| все
кц
```

23. Строка **лит** a составлена только из 0 и 1. Чему будет равно n после выполнения команд:

а)
```
n:=0
нц для i от 1 до длин (a)
| n:=n*10
| если a[i] =1
| | то n:=n+1
| все
кц
```

б)
```
n:=0
нц для i от 1 до длин (a)
| n:=n*2
| если a[i] =1
| | то n:=n+1
| все
кц
```

24. Опишите порядок закраски клеток при выполнении алгоритма "закрасить область" (А89) в трех ситуациях (рис. 69).

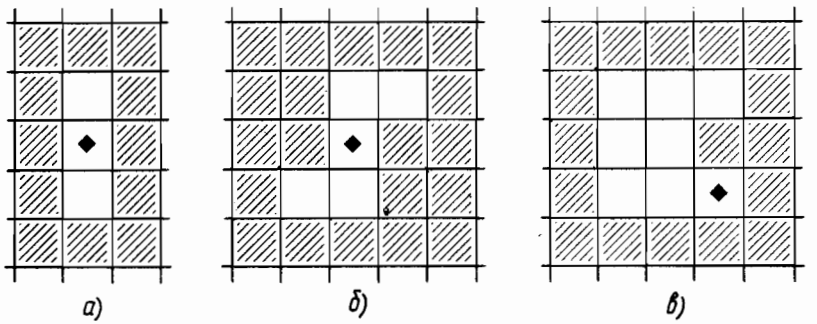


Рис. 69

25. Опишите, что произойдет при попытке выполнить алгоритм "закрасить область" (А89) на пустом поле (когда нет не только стен, но и ни одной закрашенной клетки).

26*. На поле Робота стен нет. Двигаясь из исходной клетки по незакрашенным, Робот может попасть лишь в конечное число клеток. Составьте алгоритм, вычисляющий минимальное число шагов Робота от исходной клетки до ближайшей закрашенной.

УПРАЖНЕНИЯ НА ПОВТОРЕНИЕ

1. Робот находится в левом верхнем углу огороженного со всех сторон прямоугольника. Внутри прямоугольника расположены стены, все они идут с запада на восток, глухих стен от края до края нет (рис. 70).

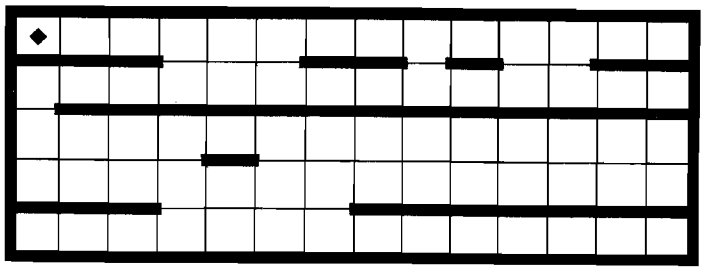


Рис. 70

а) Составьте алгоритм, перемещающий Робота в левый нижний угол прямоугольника.

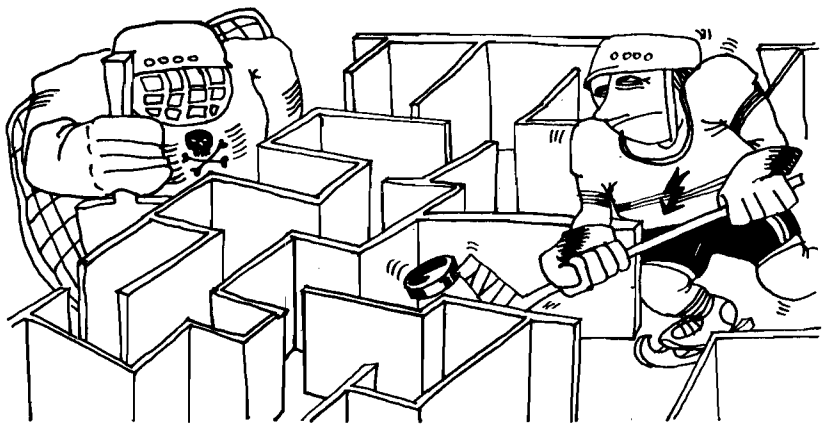
б) Составьте алгоритм, который с помощью Робота исследует расположение стен в прямоугольнике и рисует с помощью Чертежника сам прямоугольник и расположение стен в нем.

В упражнениях 2—8 постарайтесь составить алгоритм, не используя таблиц.

2. Составьте алгоритм, находящий наименьший делитель данного натурального числа, больший 1.

3. Составьте алгоритм, проверяющий, является ли данное число простым.

4. Решите упражнение 3 так, чтобы при выполнении алгоритма количество действий не превышало $C\sqrt{n}$, где n — проверяемое число, а C — некоторая не зависящая от n константа.



5. Составьте алгоритм, печатающий разложение натурального числа n на простые сомножители.

6*. Составьте алгоритм, который по заданным натуральным числам a и b находит их наибольший общий делитель c и такие целые числа x и y, что $c = a*x + b*y$. (Из анализа алгоритма должно следовать, что такие x и y существуют.)

7. Составьте алгоритм, который для заданного натурального числа n печатает все цифры его десятичной записи, начиная а) с младших, б) со старших разрядов.

8. Составьте алгоритм, который по целому положительному числу n печатает друг за другом цифры периода десятичной записи дроби 1/n. (Длина периода должна быть по возможности наименьшей.)

9. Составьте алгоритм, который определяет, станет ли строка **лит** а "перевертышем" после вычеркивания из нее всех пробелов.

10. Составьте алгоритм, который заносит в таблицу **цел таб** с [1:1000] первые 1000 натуральных чисел, делящихся нацело на 13 или на 17.

11. Элементы таблицы **вещ таб** a [1:n] расположены в порядке неубывания. Составьте алгоритм, который определяет количество различных элементов в этой таблице.

12. Составьте алгоритм, который определяет количество различных элементов в произвольной таблице.

13. Элементы таблиц **вещ таб** a [1:m], b [1:n] расположены в порядке неубывания. Составьте алгоритм, результатом которого является неубывающая таблица с [1:m+n], в которой каждый элемент встречается столько раз, сколько он встречается в таблицах a и b, вместе взятых. Количество действий не должно превышать $C \cdot (m+n)$, где C — некоторая константа.

14. Составьте алгоритм, аргументами которого являются три неубывающие таблицы, про которые известно, что существует хотя бы одно число, встречающееся в каждой из таблиц. Результатом работы алгоритма должно быть наименьшее из таких чисел. Количество действий не должно превосходить некоторой константы, умноженной на общее количество элементов во всех трех таблицах.

15. В прямоугольной таблице **вещ таб** a [1:m, 1:n] элементы в каждой строке и в каждом столбце не убывают. Составьте алгоритм, который узнает, есть ли в этой таблице число x. Количество действий не должно превышать $C \cdot (m+n)$.

16. Элементы таблиц **вещ таб** a [1:m], b [1:n] расположены в порядке неубывания. Составьте алгоритм, который выясняет, можно ли из a вычеркнуть некоторые элементы так, чтобы получилось b. Количество действий не должно превышать $C \cdot (m+n)$.

17. Даны две таблицы a [1:m] и b [1:n]. Составьте алгоритм, который определяет максимальное число k такое, что путем вычеркивания некоторых элементов из a и из b можно получить

одинаковые последовательности чисел длины k. Количество действий не должно превышать $C \cdot m \cdot n$.

18. Дана таблица a [1:n]. Составьте алгоритм, который находит минимальное k такое, что путем вычеркивания k элементов из a можно получить неубывающую последовательность чисел.

19. Составьте алгоритм, выводящий на экран первые n чисел, которые не имеют простых делителей, кроме 2, 3 и 5 (т. е. 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...). Количество действий не должно превышать $C \cdot n$.

20. Перестановкой k чисел называется последовательность, в которой каждое из чисел 1, 2, ..., k встречается ровно один раз. Составьте алгоритм, который выводит на экран все перестановки из k чисел.

21. Составьте алгоритм, который выводит на экран все возможные расстановки n ферзей на квадратной доске со стороной n, при которых никакие два ферзя не бьют друг друга.

22. Составьте алгоритм подсчета числа способов, которыми можно уплатить k рублей купюрами в 1, 3, 5, 10, 25, 50 и 100 рублей (сами способы перечислять не требуется). Количество действий не должно превышать $C \cdot k^2$.

23. Некоторые натуральные числа могут быть представлены в виде суммы кубов двух целых неотрицательных чисел (например, $9 = 8 + 1$, $27 = 27 + 0$). Составьте алгоритм, который находит наименьшее натуральное число, имеющее два разных таких представления ($8 + 1$ и $1 + 8$ — одно и то же представление).

24. В некоторой стране имеется n городов, соединенных авиационным сообщением. Стоимость билета из i-го города в j-й — положительное вещественное число c [i, j]. Составьте алгоритм, который для данных i и j определяет стоимость самого дешевого маршрута из i в j (возможно, с пересадками). Число действий не должно превышать $C \cdot n^2$.

25. Даны две строки a и b. Составьте алгоритм, который определяет, можно ли из букв, входящих в a, составить b. (Буквы можно переставлять, но каждую букву можно использовать не более одного раза.)

26. Составьте алгоритм, который по заданным вещественным числам a, b, c, d вычисляет выражения $a*c + b*d$ и $a*d - b*c$ так, чтобы выполнялись только операции сложения, вычитания и умножения, причем операция умножения выполнялась не более трех раз.

27. Робот стоит в клетке А, вправо от которой отходит петляющий коридор шириной в одну клетку (рис. 71). Составьте алгоритм, выводящий Робота из коридора (в клетку Б).

28. Робот находится в левом верхнем углу внутри прямоугольника, огороженного стенами. Составьте алгоритм, который закрашивает весь прямоугольник и возвращает Робота в исходное положение за минимальное число команд Роботу.

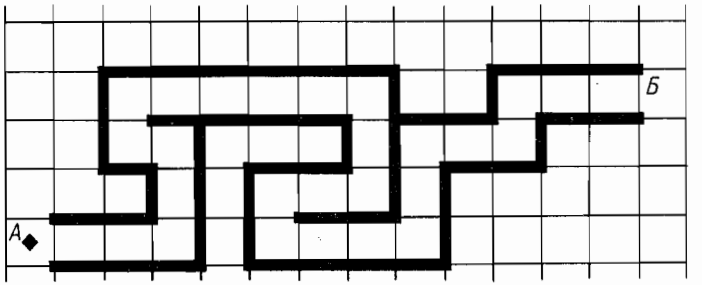


Рис. 71

29. Составьте алгоритм, который определяет положение биллиардного шара, выпущенного из угла квадратного биллиарда размером $d \times d$ м со скоростью v м/с под углом α к одной из сторон через t с после начала движения (трения нет, удары упругие).

30. Стены на поле Робота образуют замкнутую линию — "забор". Робот стоит, "прислонившись" к забору, т. е. в одной из клеток, примыкающих к забору вплотную. Составьте алгоритм, который определяет, внутри или снаружи забора стоит Робот (после выполнения алгоритма Робот должен оказаться в исходном положении).

31*. Стены на поле Робота образуют замкнутую линию — "забор". Робот стоит, "прислонившись" к забору: а) изнутри; б) снаружи. Составьте алгоритм, который определяет количество огороженных забором клеток ("площадь участка").

32. Придумайте систему команд исполнителя "Строитель", умеющего переносить прямоугольные блоки по клетчатой строительной площадке (фото 21 вклейки). Составьте алгоритм, при выполнении которого Строитель огораживает забором прямоугольник размером 8 на 16 клеток.

33. Придумайте систему команд исполнителя "Вездеход", умеющего поворачиваться, определять расстояние до ближайшего препятствия впереди и двигаться на заданное расстояние вперед (фото 22 вклейки). Составьте алгоритм определения числа поворотов дороги, по которой едет Вездеход.

34. Двуног — это существо с тяжелой головой на очень длинной шее и двумя ногами с присосками (фото 20 вклейки). Двуног умеет независимо поворачивать шею и ноги. Сильно наклонив шею, Двуног может начать падать и, подставив ногу,— сделать шаг. Придумайте систему команд Двунога и составьте алгоритмы для ходьбы: а) по ровному месту; б) по лестнице.

ГЛАВА 2. УСТРОЙСТВО ЭВМ

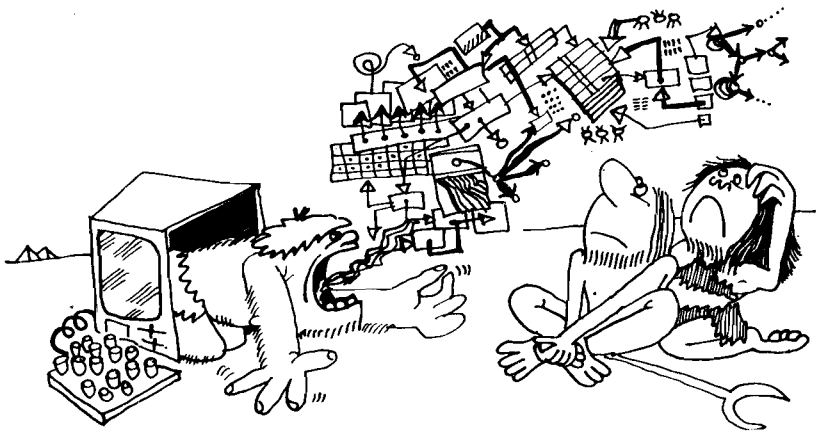
§ 17. ФИЗИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

17.1. КОДИРОВАНИЕ ИНФОРМАЦИИ ЭЛЕКТРИЧЕСКИМИ СИГНАЛАМИ

Как вы уже знаете, информация хранится и обрабатывается в ЭВМ в двоичном виде — в виде последовательностей нулей и единиц. В современных ЭВМ широко распространено представление ''1'' высоким напряжением (например, +5 Вольт), а ''0'' — низким (около 0 Вольт).

17.2. ЭЛЕКТРОННЫЙ КЛЮЧ

Основным элементом современных ЭВМ является транзистор. В радиоаппаратуре транзисторы чаще всего используются для усиления плавно меняющихся сигналов. В вычислительной же технике транзисторы используются в так называемом ***ключевом*** режиме. В этом режиме транзистор можно представлять как обычный выключатель, который в одном положении проводит ток (замкнут), а в другом — нет (разомкнут). В отличие от бытового выключателя, однако, включение и выключение транзистора производится также с помощью электричества.



Схемное обозначение МОП-транзисторов

а) p-канального б) n-канального

Рис. 72

Существует много типов транзисторов. Мы рассмотрим только ***МОП-транзисторы*** (рис. 72; сокращение ''МОП'' — Металл-Оксид-Полупроводник — относится к устройству транзистора). Каждый МОП-транзистор имеет четыре контакта: сток, исток, затвор и подложку. N-канальный транзистор (рис. 72, *б*) проводит ток между истоком и стоком (замкнут), только если на затвор подано положительное (относительно подложки) напряжение. P-канальный транзистор, наоборот, проводит ток только если напряжение на затворе (рис. 72, *а*) меньше напряжения на подложке.

17.3. ВЕНТИЛЬ ''НЕ''

В качестве элементарных строительных ''кубиков'' в ЭВМ обычно используются не сами транзисторы, а более крупные элементы, называемые ***вентилями***. Существуют разные способы (технологии) объединения транзисторов в вентили.

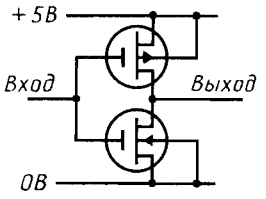


Рис. 73

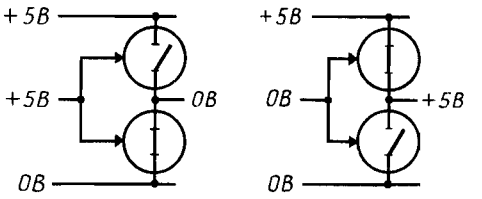


Рис. 74

Рассмотрим так называемый КМОП-вентиль ''не'' (рис. 73). Если на его вход подать напряжение +5В (''1''), то нижний (n-канальный) транзистор замкнется, а верхний будет разомкнут (рис. 74, *а*) — на выходе вентиля будет напряжение 0В (''0'').

Если же на вход вентиля подать 0В, то замкнется верхний (p-канальный) транзистор, поскольку напряжение на его затворе (0В) будет меньше напряжения на подложке (+5В). На нижнем же транзисторе напряжения на затворе (0В) и подложке (0В) будут совпадать и транзистор будет разомкнут (рис. 74, *б*). В результате на выходе вентиля будет +5В (''1'').

Вход	Нижний транзистор			Верхний транзистор			Выход
	затвор	подложка	состояние	затвор	подложка	состояние	
+5В	+5В	0В	замкнут	+5В	+5В	разомкнут	0В
0В	0В	0В	разомкнут	0В	+5В	замкнут	+5В

Таким образом, рассмотренный вентиль является простейшим электронным устройством обработки информации — он инвертирует (''переворачивает'') бит: ''1'' (+5В) на входе превращает в ''0'' (0В) на выходе, а ''0'' превращает в ''1''. Если сопоставить ''1'' значение **да**, а ''0'' — **нет**, то работу этого вентиля можно описать формулой

выход = **не** вход

Поэтому рассмотренный вентиль называется инвертором или вентилем ''не''.

17.4. ВЕНТИЛЬ "ИЛИ-НЕ"

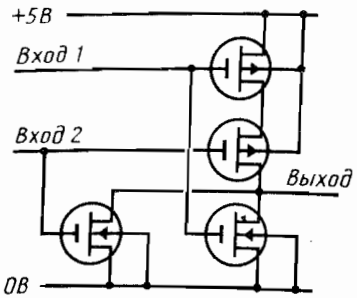


Рис. 75

В ЭВМ используются также другие типы вентилей, соответствующие другим элементарным операциям по обработке информации. На рисунке 75 изображен КМОП-вентиль для формулы

выход = **не** (вход1 **или** вход2)

Состояния транзисторов (замкнут/разомкнут) и значение на выходе вентиля ''или-не'' для всех сочетаний значений входов изображены на рисунке 76:

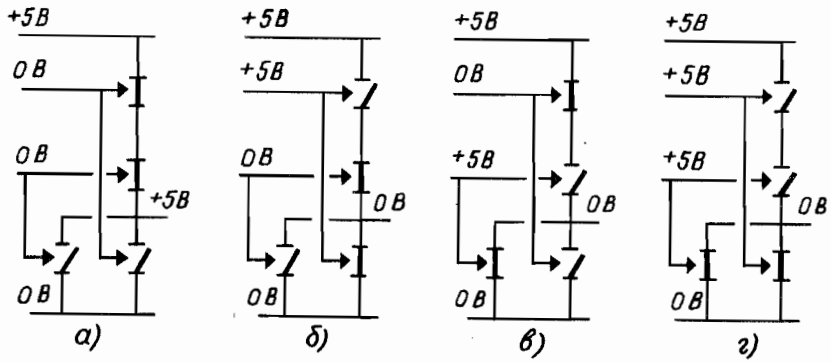


Рис. 76

17.5. ОБОЗНАЧЕНИЯ ВЕНТИЛЕЙ

В электронике наиболее широко распространены вентили пяти типов:

а) ''и'' : выход = вход1 **и** вход2;
б) ''или'' : выход = вход1 **или** вход2;
в) ''не'' : выход = **не** вход;
г) ''и-не'' : выход = **не** (вход1 **и** вход2);
д) ''или-не'' : выход = **не** (вход1 **или** вход2).

Чтобы не загромождать схемы лишними деталями, для изображения вентилей применяются специальные обозначения (рис. 77).

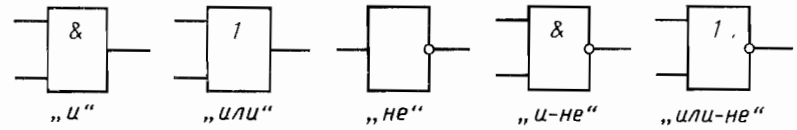


Рис. 77

Приведенный набор вентилей является избыточным — одни вентили в нем выражаются через другие. Используя только вентили типа ''или-не'', например, можно собрать любую схему, в частности, вентиль ''и'' (рис. 78), заданный формулой ''выход = вход1 **и** вход2''.

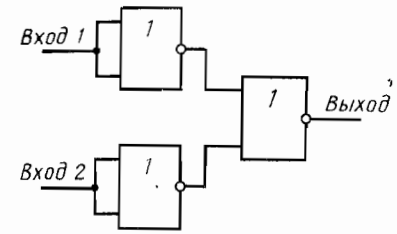


Рис. 78

17.6. ПРОЦЕССОР

Обработку информации в ЭВМ производит процессор, который выполняет команды программы, хранящиеся в памяти ЭВМ. Эти команды выполняются специальными электронными схемами, состоящими из десятков и сотен тысяч вентилей.

Обычно команда обрабатывает информацию не по одному биту, а одновременно группами по 8, 16, 32 или 64 бита.

Число одновременно обрабатываемых битов называется ***разрядностью процессора*** и является одной из важнейших характеристик ЭВМ.

Вместе с ***быстродействием*** (числом выполняемых команд в секунду) разрядность характеризует объем информации, перерабатываемой процессором ЭВМ в единицу времени.

17.7. ЭЛЕМЕНТ ПАМЯТИ ТРИГГЕР

Приведенные выше примеры демонстрировали возможность обработки информации на ЭВМ. Покажем теперь, как можно информацию запоминать. Поскольку любая информация в ЭВМ представляется в двоичном виде, рассмотрим запоминание и хранение элементарной порции информации — одного бита. Электронная схема, запоминающая один бит информации, называется ***триггером***.

Рассмотрим простейший так называемый RS-триггер (рис. 79).

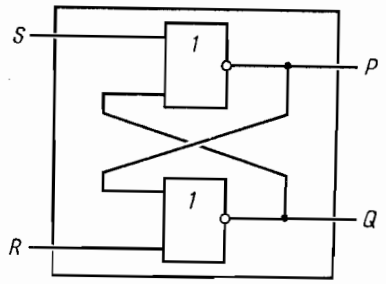


Рис. 79

Если на входы этого триггера подать S=1, R=0, то (независимо от состояния Q) на выходе P верхнего вентиля появится 0. После этого на входах нижнего вентиля окажется R=0, P=0 и выход Q станет равным 1.

Если теперь перестать подавать сигналы на триггер (S=0, R=0), то, поскольку входами верхнего вентиля являются S=0 и Q=1, его выход P останется 0. Аналогично, поскольку входами нижнего вентиля являются R=0 и P=0, его выход Q по-прежнему будет 1. Таким образом, установленные значения выходов P и Q не изменятся при переходе к S=0, R=0.

Точно так же при подаче S=0, R=1 на выходах появятся Q=0, P=1 и эти значения выходов сохранятся при снятии "1" со входа R (R=0, S=0).

Таким образом, при S=0, R=0 триггер может находиться в двух разных состояниях: Q=1 и Q=0. Выход Q и является значением запомненного бита.

Вход S	Вход R	Действие триггера	Выход Q
1	0	запоминание 1	1
0	1	запоминание 0	0
0	0	хранение бита	запомненный бит

Поскольку один триггер запоминает один бит, то для запоминания байта (8 бит) нужно 8 триггеров, для запоминания килобайта — 1024 * 8=8192 триггера и т. д. Современные микросхемы памяти объемом менее 1 см^3 способны запоминать миллионы бит информации.

17.8. ПАМЯТЬ

В памяти ЭВМ запоминающие элементы обычно объединяют в группы из нескольких (8, 16, 32, 64) бит. В ЭВМ УКНЦ, например, используются группы из 8 бит, которые занумерованы числами от 0 до 65535. Такие числа называются ***адресами*** байтов. Для кодирования адреса в УКНЦ используются последовательности из 16 бит. Число бит, с помощью которых кодируется адрес, называется ***разрядностью адреса*** ЭВМ. Это число характеризует максимальный объем памяти (информации), одновременно доступный процессору.

17.9. ВЗАИМОДЕЙСТВИЕ ПРОЦЕССОРА И ПАМЯТИ

Процесс взаимодействия процессора и памяти сводится в основном к двум операциям: ***запись*** информации в память и ***чтение*** информации из памяти. При записи процессор по специальным проводникам (они называются ***шиной адреса***) передает биты, кодирующие адрес; по другим проводникам передает сигнал "запись" и по еще одной группе проводников (она называется ***шиной данных***) передает записываемую информацию. Конечно, слово "передает" означает просто, что схемы процессора подают на проводники напряжения, соответствующие "0" и "1". При чтении процессор устанавливает сигналы на шине адреса и считывает информацию с шины данных.

Число одновременно передаваемых по шине адреса и шине данных разрядов (битов) называется ***разрядностью*** соответствующей шины и является важной характеристикой ЭВМ. Разрядность шины адреса определяет максимальное общее количество доступной памяти; разрядность шины данных — максимальную порцию информации, которую можно получить из памяти за один раз.

В широко распространенном персональном компьютере IBM PC, например, процессор Intel 8088 является 16-разрядным, адрес — 16-разрядным, шина адреса — 20-разрядной, а шина данных — 8-разрядной. Поэтому для получения 16 разрядов данных, которые процессор может обработать за одну команду, он должен обратиться к памяти дважды.

17.10. ПОКОЛЕНИЯ ЭВМ

История развития ЭВМ делится на отрезки, называемые поколениями. Смена поколений связана прежде всего с изменением

физических принципов работы и технологией производства элементов, входящих в ЭВМ.

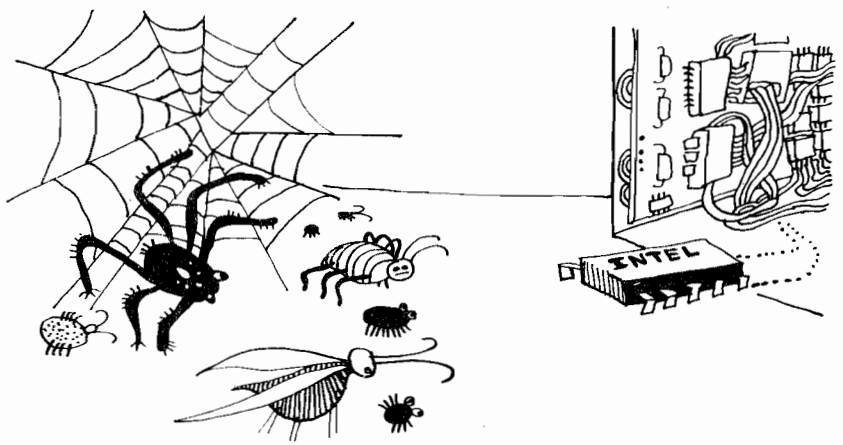
ЭВМ ***первого поколения*** использовали электронные лампы (радиолампы) — те самые, которые еще можно найти в некоторых телевизорах. Помимо ламп, ЭВМ включали в себя другие элементы (резисторы, конденсаторы и т. п.). В настоящее время ЭВМ первого поколения не применяются.

Элементную базу ЭВМ ***второго поколения*** составляли транзисторы, изготовлявшиеся в виде отдельных деталей (каждый в своем корпусе). Они занимали меньше места, чем радиолампы, потребляли меньше энергии и были более надежны. Это позволило сделать ЭВМ более компактными, дешевыми и экономичными. Ко второму поколению ЭВМ относится одна из наиболее известных советских ЭВМ БЭСМ-6, которая используется до сих пор (хотя уже не выпускается).

Третье поколение ЭВМ обязано своим возникновением революционной идее изготовить фрагмент электронной схемы, содержащий много транзисторов, резисторов, конденсаторов, проводников и т. д., в виде участков различных веществ на поверхности одной полупроводниковой (обычно кремниевой) пластинки размером с клеточку школьной тетради. Такие электронные устройства получили название ***интегральных схем*** (от лат. integrare — собирать вместе, объединять) или ***микросхем***.

Одними из первых ЭВМ третьего поколения были машины американской фирмы IBM. Аналогичные им советские ЭВМ серии ЕС выпускаются до сих пор.

Дальнейший прогресс состоял прежде всего в отработке технологии размещения все большего и большего числа вентилей в одной микросхеме. Это число называется ***степенью интеграции***. Для первых интегральных схем оно было невелико (единицы и десятки вентилей). Постепенно степень интеграции увеличивалась и сейчас достигает сотен тысяч вентилей.



Вехой на пути совершенствования интегральных схем стал момент, когда удалось сделать в одной микросхеме целый процессор ЭВМ. Появление в 1971 г. таких процессоров, названных ***микропроцессорами***, и ознаменовало переход к ***четвертому поколению*** ЭВМ. В результате ЭВМ стали еще дешевле и надежнее. Примерами микропроцессорных ЭВМ служат школьные ЭВМ Корвет и УКНЦ.

В ЭВМ Корвет процессор — это микросхема К580ВМ80А отечественного производства (аналог одного из первых в мире микропроцессоров Intel 8080 — США, 1976 г.). В ЭВМ УКНЦ использована отечественная микросхема К1801ВМ2 (система команд совпадает с системой команд ЭВМ серии PDP-11, выпускавшихся фирмой "Digital Equipment Corporation" (США) с 1970 г.).

17.11. ИЗГОТОВЛЕНИЕ МИКРОСХЕМ

При изготовлении микросхем отнюдь не приходится последовательно изготовлять каждый транзистор. С помощью процесса, похожего на печать фотокарточек, в результате одной серии технологических операций на полупроводниковой пластине одновременно формируют сотни тысяч нужным образом соединенных вентилей. Другими словами, примеси нужных веществ, слои металла и изолятора наносятся одновременно во все необходимые места пластинки.

Более того, в процессе "печати" нужные примеси наносятся на поверхность не одной микросхемы, а на поверхность большой пластины (диаметром около 10 см), которая после обработки распиливается на сотни отдельных микросхем.

Таким образом, миллионы транзисторов, соединенных в сотни сложных микросхем, изготавливаются всего за десяток операций, в ходе которых на поверхность пластины наносятся разные вещества.

Размеры деталей современных микросхем составляют несколько микрон (в десятки раз меньше толщины человеческого волоса). Поэтому мельчайшие загрязнения в процессе производства (например, пылинки в воздухе или даже водяные пары от дыхания человека) приводят к изготовлению бракованных микросхем. Поскольку починить бракованную микросхему невозможно, их приходится выбрасывать. Процент работающих микросхем называется ***выходом годных*** и может изменяться от 1—2% до 95%. Величина выхода годных зависит от сложности и типа микросхем и дает представление о культуре и технологии производства.

УПРАЖНЕНИЯ

1. Нарисуйте схему вентиля "и-не" из МОП-транзисторов, соответствующую формуле "выход=**не** (вход1 **и** вход2)".

2. Нарисуйте схему из вентилей "или-не", соответствующую формуле "выход = **не** (вход1 **и** вход2)". Сравните число транзисторов в этой схеме с числом транзисторов в вашем решении упражнения 1.

3. Нарисуйте схему из вентилей, изображенных на рисунке 77, соответствующую формуле:

а) выход = (вход1 ≠ вход2);
б) выход = (вход1 = вход2);
в) выход = (вход1 **и** вход2 **и** вход3).

4. Нарисуйте схему только из вентилей "и-не", соответствующую формуле:

а) выход = **не** вход;
б) выход = вход1 **и** вход2;
в) выход = вход1 **или** вход2;
г) выход = (вход1 = вход2).

5. Решите упражнение 4, используя только вентили "или-не".

6. Что будет на выходах P и Q триггера (рис. 79) при R = 1, S = 1?

7. Замените в схеме триггера (рис. 79) вентили "или-не" на вентили "и-не". Опишите работу получившегося триггера.

8. Придумайте не менее трех разных способов кодирования целых чисел в диапазоне от 0 до 255 последовательностями битов одинаковой длины.

9. Придумайте какой-нибудь способ кодирования битами целых чисел от 0 до 7. Используя вентили (рис. 77), нарисуйте схему, которая по кодам двух чисел в диапазоне:

а) от 0 до 1;
б) от 0 до 3

выдает код их суммы.

10. Считая, что электронная лампа (аналог транзистора) сравнима по размерам с обычной осветительной лампочкой, оцените, какой размер имел бы 1 мегабит памяти в ЭВМ первого поколения.

§ 18. КОМАНДЫ И ОСНОВНОЙ АЛГОРИТМ РАБОТЫ ПРОЦЕССОРА

ЭВМ может выполнять сложные алгоритмы, перерабатывая огромные объемы информации. Эта сложнейшая работа состоит из громадного числа "электронных переключений", которые выполняются в ЭВМ в соответствии с заданной программой. В этом параграфе мы проанализируем работу ЭВМ на более высоком уровне: познакомимся с тем, что такое машинная программа, какие элементарные команды (операции) умеет выполнять ЭВМ и, главное, как обеспечивается автоматизм работы процессора. Если сравнить транзисторы и триггеры с песчинками в стенах здания ЭВМ, то теперь мы познакомимся с кирпичиками, из которых оно сложено.

18.1. ПАМЯТЬ, ПРОЦЕССОР, ПРОГРАММА

Напомним, что информация в памяти ЭВМ кодируется последовательностями нулей и единиц, т. е. с помощью битов. Группа из 8 битов называется байтом. Память ЭВМ в целом можно представлять себе как линейную таблицу байтов. Байты в памяти ЭВМ нумеруются с нуля: 0, 1, 2, 3, ... Номер байта называется его ***адресом***.

Выполняемая ЭВМ ***машинная программа***, как и любая другая информация, кодируется некоторой последовательностью байтов и размещается в памяти. Адрес первого байта программы называется ***адресом программы***. Программа состоит из ***машинных команд*** (или просто команд). Как мы уже говорили, ЭВМ выполняет лишь очень простые команды: сложение двух чисел, сравнение чисел и т. п.

Важнейшим устройством ЭВМ является ***процессор***. Именно процессор команда за командой выполняет программы. Процессор имеет небольшую собственную память. Часть этой памяти называется ***Счетчиком Команд*** (или просто СК) и содержит неотрицательное целое число — адрес следующей машинной команды, которую будет выполнять процессор. Еще одна часть собственной памяти процессора называется ***Слово Состояния Процессора*** (ССП). В частности, при выполнении всех команд сложения, вычитания, сравнения и пр. в ССП запоминается знак S результата команды:

если результат меньше нуля, то S = −1,
если результат равен нулю, то S = 0,
если результат больше нуля, то S = +1.

18.2. ОСНОВНОЙ АЛГОРИТМ РАБОТЫ ПРОЦЕССОРА

Процессор начинает работу после того, как программа записана в память ЭВМ, а в Счетчик Команд записан адрес первой команды программы. Работу процессора можно описать следующим циклом:

нц
| чтение команды из памяти по адресу, записанному в СК
| увеличение СК на длину прочитанной команды
| выполнение прочитанной команды
кц

Обратите внимание, что после чтения очередной команды процессор увеличивает СК на длину команды. Поэтому при следующем выполнении тела цикла процессор прочтет и выполнит следующую команду программы, потом еще одну и т. д. Цикл закончится, когда встретится и будет выполнена специальная команда "стоп". В итоге ЭВМ автоматически, без участия человека, команда за командой выполнит ***всю программу*** целиком.

Автоматизм работы процессора, возможность выполнения длинных последовательностей команд без участия человека — одна из основных отличительных особенностей ЭВМ как универсальной машины обработки информации.

18.3. ПРИМЕРЫ КОМАНД ПРОЦЕССОРА

Поясним принцип работы процессора на примере нескольких команд школьной ЭВМ "Электроника УКНЦ". Каждая команда занимает в памяти 1, 2 или 3 слова (***словом*** называется пара байт с адресами 0—1, 2—3, 4—5 и т. д.). Целое число занимает в памяти ЭВМ также 1 слово.

Команда	Длина в словах	Что происходит при выполнении команды
стоп	1	ЭВМ кончает работу
очистить x	2	x:=0
увеличить x на единицу	2	x:=x+1
уменьшить x на единицу	2	x:=x−1
переслать из x в y	3	y:=x
прибавить x к y	3	y:=y+x
вычесть x из y	3	y:=y−x
сравнить x с y	3	S:=sign (x−y)
сравнить x с нулем	2	S:=sign (x)
если меньше переход на d	1	проверяется знак S в ССП. Если указанное в команде условие выполнено, то СК:=СК+2*d иначе СК не меняется
если равно переход на d	1	
если больше переход на d	1	
если не меньше переход на d	1	
если не равно переход на d	1	
если не больше переход на d	1	
переход на d	1	

Конечно, мы приводим лишь малую толику команд ЭВМ "Электроника УКНЦ". И даже для этих команд мы рассматриваем только один простейший вариант (в то время как некоторые команды имеют свыше 100 различных модификаций!). Тем не менее это реальные команды реальной ЭВМ.

18.4. ПРИМЕР ПРОСТЕЙШЕЙ МАШИННОЙ ПРОГРАММЫ

Пусть в памяти по адресам 100 и 102 расположены два целых числа, а требуется вычислить сумму чисел и поместить ее в

память по адресу 104. Это можно сделать с помощью двух команд:

переслать из 100 в 104
добавить 102 к 104

Чтобы заставить процессор выполнить эти две команды, их надо поместить в память, например, начиная с адреса 0, поместить вслед за ними команду "стоп" и записать в СК адрес первой команды, т. е. 0. Предположим, что это сделано и что в начальный момент в памяти по адресу 100 записано число 11, по адресу 102 — число 15, а по адресу 104 — число 7:

	Память	СК
0	переслать	0
2	100	
4	104	
6	добавить	
8	102	
10	104	
12	стоп	
100	11	
102	15	
104	7	

После выполнения этой программы получим

	Память	СК
100	11	14
102	15	
104	26	

И команды и числа изображены на рисунках в привычном для человека виде. Напомним, что в ЭВМ они кодируются последовательностями из 0 и 1. Как именно, нам неважно, но для полноты картины мы один раз такую кодировку покажем:

Адрес	Содержимое памяти ЭВМ	Команда
0	0 001 011 111 011 111	переслать
2	0 000 000 001 100 100	100
4	0 000 000 001 101 000	104
6	0 110 011 111 011 111	добавить
8	0 000 000 001 100 110	102
10	0 000 000 001 101 000	104
12	0 000 000 000 000 000	стоп

18.5. КОМАНДЫ УСЛОВНОГО И БЕЗУСЛОВНОГО ПЕРЕХОДА

Среди команд процессора есть команды, которые изменяют значение Счетчика Команд. Такие команды называются командами ***перехода***. Команды переходов меняют или не меняют СК в зависимости от величины S, запомненной в ССП. Например, команда "если меньше переход на d" выполняется так:

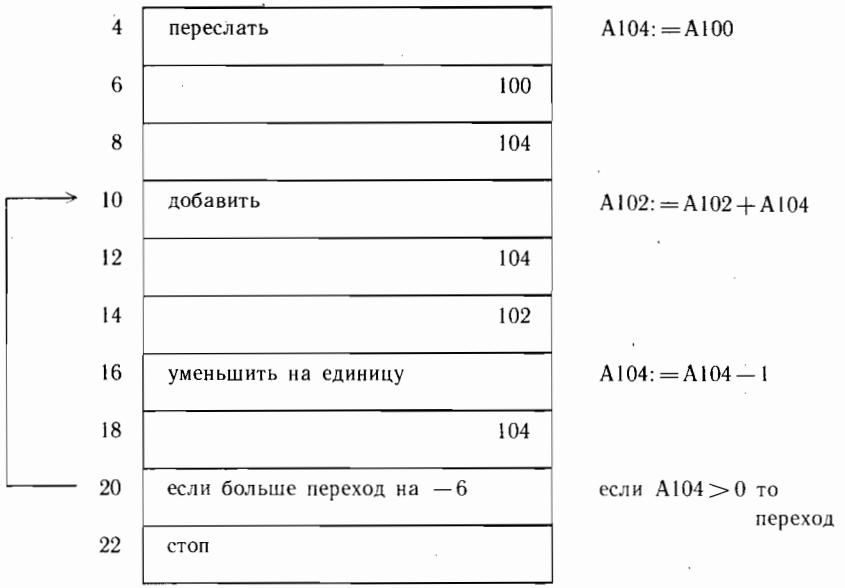
если S = −1 **то** СК := СК + 2*d **все**

Заметим, что процессор выполняет команду после увеличения СК на длину команды. Поэтому, если условие не выполнено, процессор просто перейдет к выполнению следующей команды. Если же условие выполнено, то процессор либо пропустит d слов (2*d байтов) в памяти (при положительном d), либо вернется на d слов назад (при отрицательном d).

Существует также команда ***безусловного перехода*** "переход на d", которая увеличивает СК на 2*d независимо ни от чего.

18.6. МАШИННАЯ РЕАЛИЗАЦИЯ ЦИКЛА ПОКА

Уменьшив содержимое СК, можно заставить процессор повторить некоторую последовательность команд (и таким образом реализовать цикл). Рассмотрим, например, машинную программу, которая помещает в память по адресу 102 сумму чисел K + (K − 1) + ... + 2 + 1, где K — число, записанное по адресу 100:



Здесь и в упражнениях A100, A102, A104 и др. обозначают содержимое слова памяти по соответствующему адресу. В приведенной программе A100 является аргументом, A102 — результатом, а A104 — промежуточной величиной.

УПРАЖНЕНИЯ

1. Напишите машинную программу, при выполнении которой процессор помещает в память по адресу 110:
а) 0;
б) 1;
в) −1;
г) −A100;
д) A100 + A102 + A104;
е) abs (A100);
ж) max (A100, A102);
з) количество положительных среди A100, A102;
и) количество максимальных среди A100, A102;
к) сумму 2K + (2K − 2) + ... + 4 + 2, где K = A100;
л) K-й член последовательности Фибоначчи, где K = A100.

2. В программе 5 команд. Может ли число шагов при выполнении этой программы оказаться равным: а) 1; б) 5; в) 100?

3. Напишите программу, которая в A104 и A106 помещает частное и остаток от деления A100 на A102.

4. Напишите программу, которая в A104 помещает наибольший общий делитель чисел A100 и A102.

§ 19. УСТРОЙСТВА ВВОДА / ВЫВОДА ИНФОРМАЦИИ

19.1. КЛАВИАТУРА

Простейшая клавиатура (например, в ЭВМ "Ямаха" и "Электроника УКНЦ") устроена так: под клавишами расположена тонкая полиэтиленовая пленка (фото 15 вклейки), склеенная из трех слоев. На верхнем и нижнем слоях нанесены металлизированные полоски (проводники), средний слой служит изолятором. В местах расположения клавиш в среднем слое сделаны небольшие отверстия, поэтому при нажатии на клавишу верхний слой прогибается, касается нижнего и замыкает контакт. Если на i-й проводник нижнего слоя (на фото левая группа проводников) подать "1" (высокое напряжение), то в j-м проводнике верхнего слоя (на фото — правая группа) "1" появится только в том случае, если клавиша на пересечении i-го и j-го проводников нажата (контакт замкнут). Меняя i и анализируя наличие или отсутствие "1" в проводниках верхнего слоя, ЭВМ может узнать, какую клавишу или группу клавиш нажал человек.

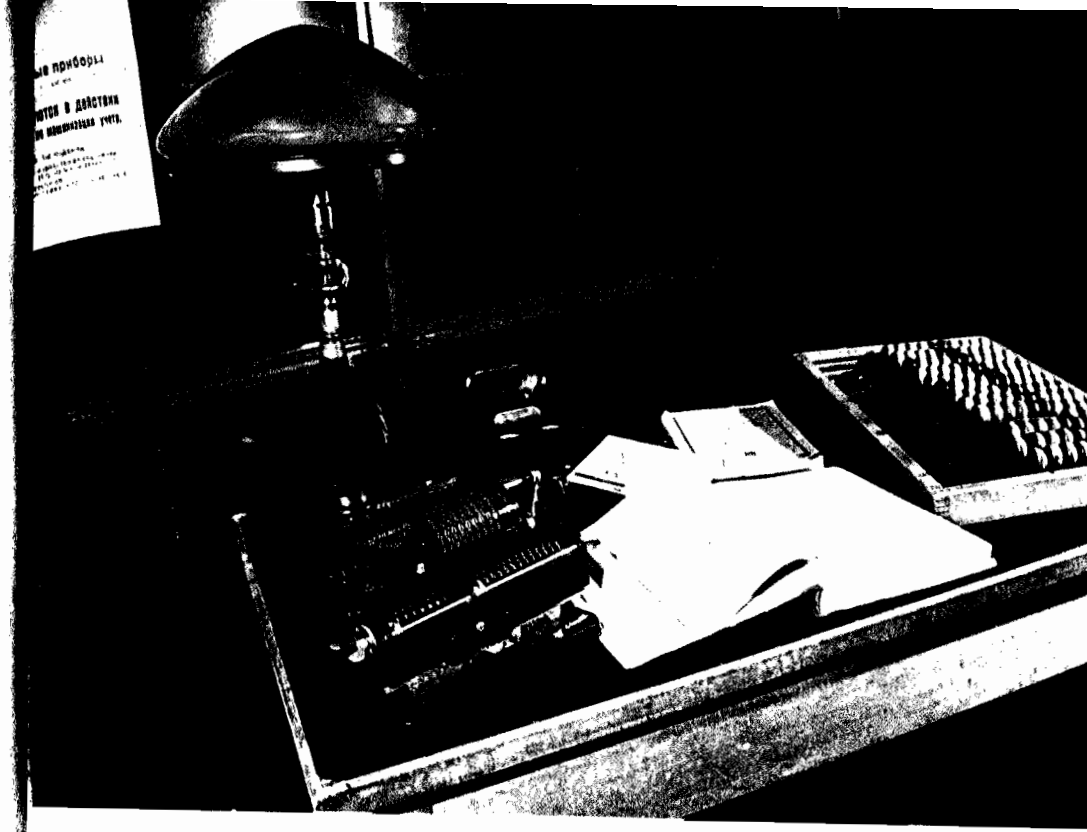
19.2. МОНИТОР

Монитор похож на обычный телевизор. Основной элемент монитора — электронно-лучевая трубка, внутри которой в вакууме расположены отрицательный электрод ("электронная пушка") и покрытый специальным веществом — люминофором — экран (положительный электрод). Под действием высоких напряжений (около 25 тыс. вольт) электроны вырываются из "пушки", разгоняются в вакууме, ударяются об экран и вызывают свечение люминофора. В зависимости от химического состава люминофоры светятся разными цветами: белым, красным, синим, зеленым, желтым.

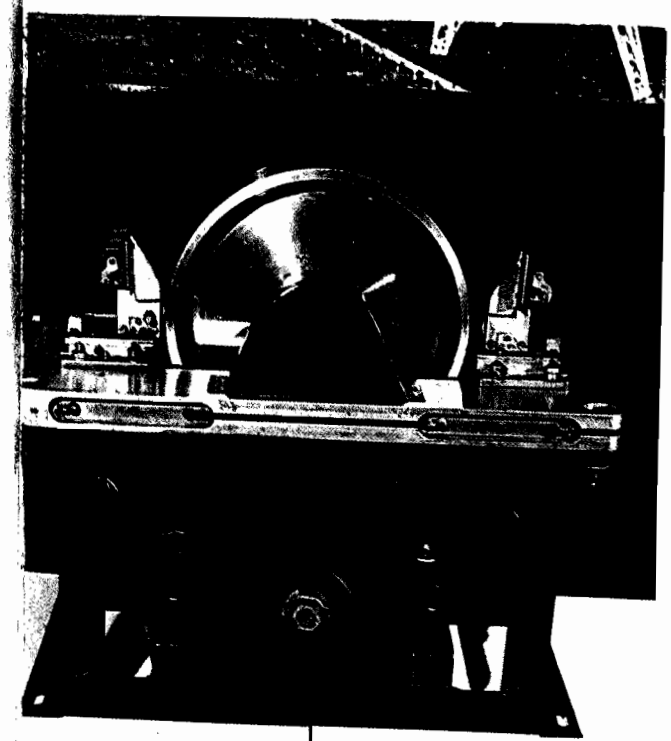
С помощью электрических полей вылетевшие из пушки электроны можно ускорить или затормозить, т. е. изменить силу удара об экран и тем самым яркость свечения люминофора.

Как и в телевизоре, вылетевшие из "пушки" электроны с помощью электрических полей фокусируются так, чтобы все они попали в одну точку (т. е. в каждый момент времени светится только одна точка экрана). Специальная отклоняющая система, однако, заставляет электронный луч быстро (несколько десятков раз в секунду) строчка за строчкой прочерчивать весь экран. Поэтому у человека создается иллюзия видимости целой картинки. Подавая в нужные моменты времени напряжение на систему управления яркостью, можно заставить светиться те или иные точки и создать на экране нужное изображение.

В цветных мониторах экран покрыт точками разных люминофоров с красным, синим и зеленым свечением, а многоцветная картинка получается наложением (совмещением) трех картинок этих базовых цветов.



1. Рабочее место счетного работника начала века, средства обработки и хранения информации: арифмометр, счеты, конторские книги



2. Магнитный барабан 50-х годов — «дедушка» современных запоминающих устройств на магнитных дисках

ВВЕДЕНИЕ

Мы начинаем изучать новый предмет - информатику. Информатика изучает методы представления, накопления, передачи и обработки информации с помощью электронно-вычислительных машин (ЭВМ). Что же такое информация, что такое ЭВМ и как ЭВМ обрабатывает информацию? Этим вопросам посвящены §1-3.

1. ИНФОРМАЦИЯ

1.1. ВЕЩЕСТВО, ЭНЕРГИЯ, ИНФОРМАЦИЯ - ВАЖНЕЙШИЕ СУЩНОСТИ НАШЕГО МИРА

Вещество - это все, что вокруг нас, это воздух и вода, леса, горы и травы, хлеб и металл. Вещество - это то, что мы едим, чем дышим, из чего шьем одежды и строим мосты, делаем те-

A>_

3. Начало текста учебника на экране ЭВМ

4. Школьный кабинет информатики на основе болгарской персональной ЭВМ «Правец»

5. Обработка текстов в многооконной системе «Микро-Мир»

6—7. Программа на Бейсике (6) и результат ее работы на экране (7)

Всего - 41 файл
Свободно 104 Кбайт

MIM	.COM	26552
MSXDOS	.SYS	2432
COMMAND	.SYS	6784
MIM88	.HLP	2974
12	.19	15381
13	.23	11361
АUTOEXEC	.BAT	18
11	.14	15431
13	.26	12271
13	.24	11482
MIM	.DIR	4312

Робот работает на клетчатом "поле" (между клетками и быть расположены стены) и умеется мелком в одной кле На бумаге, на классной доске и на экране ЭВМ мы будем и ражать поле Робота так, как показано на рисунке 4.

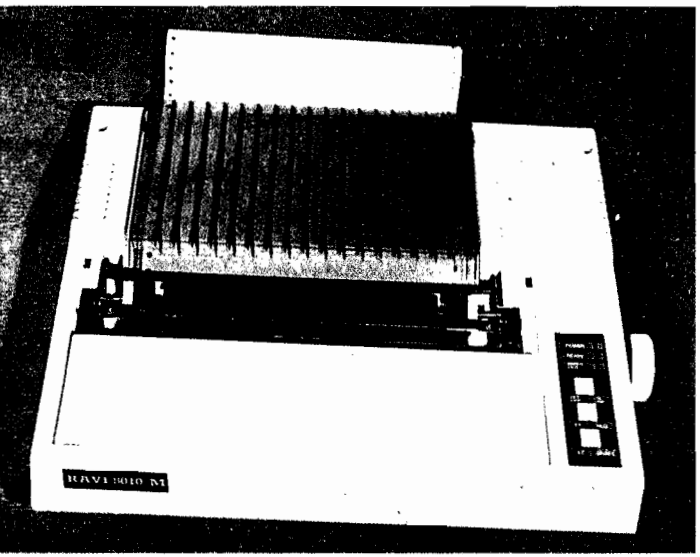
1. ИНФОРМАЦИЯ

СТВО, ЭНЕРГИЯ, ИНФОРМАЦИЯ -
ИЕ СУЩНОСТИ НАШЕГО МИРА

что вокруг нас, это воздух и вода,
б и металл. Вещество - это то, что мы

Атр: ...x... 39
285 5

```
REM:СНЕЖНАЯ БАБА
SCREEN2
CIRCLE(200,85),27,15
CIRCLE(200,150),37,15
CIRCLE(200,40),18,15
CIRCLE(168,68),10,15
CIRCLE(232,68),10,15
CIRCLE(206,35),2,15
CIRCLE(200,65),2,15
CIRCLE(200,85),2,15
CIRCLE(200,105),2,15
CIRCLE(194,35),2,15
PAINT(205,40),15
CIRCLE(200,47),8,1,3.14,6.28,1/
CIRCLE(200,42),1,8
PSET(200,42),8
LINE(165,180)-(155,30),15
LINE(166,180)-(156,30),15
LINE(155,30)-(148,15),15
LINE(156,30)-(149,15),15
```



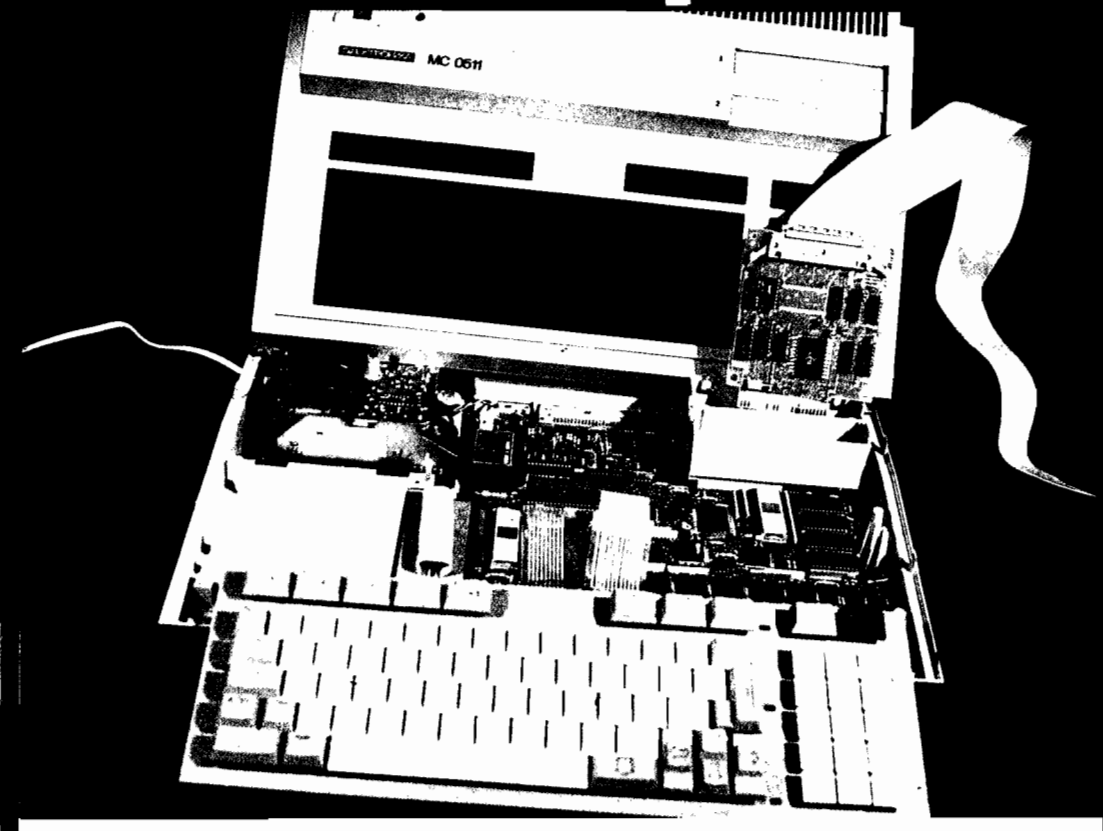
8. Серийная школьная ЭВМ «Корвет»

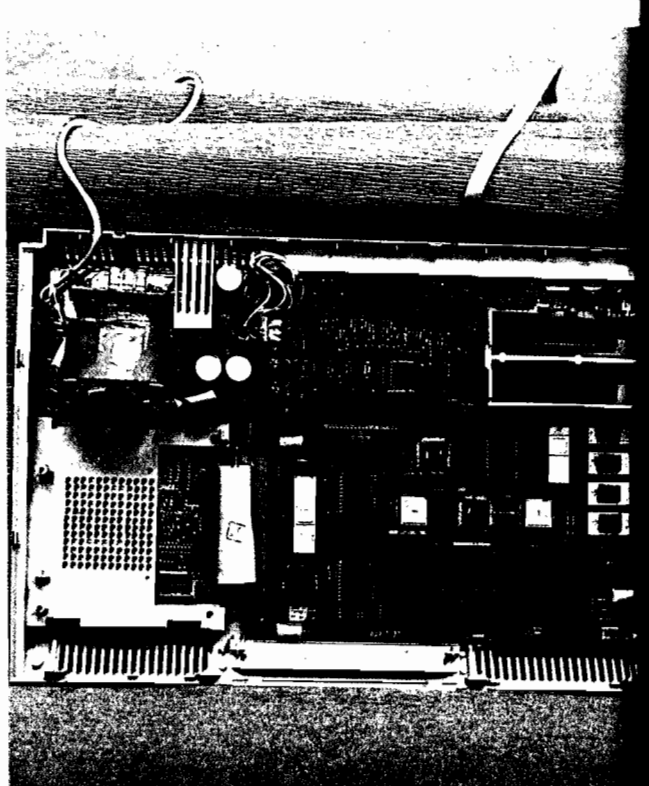
9. Принтер

10. Головка принтера крупным планом

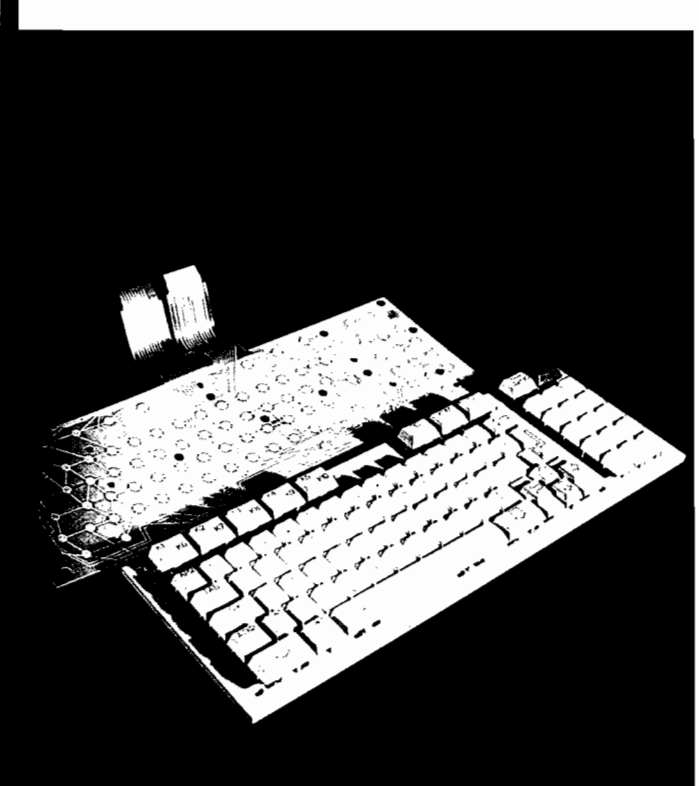
11. Электронная плата ЭВМ «Корвет» (располагается в клавиатуре)

12. Серийная школьная ЭВМ «Электроника МС-0511» (УКНЦ)



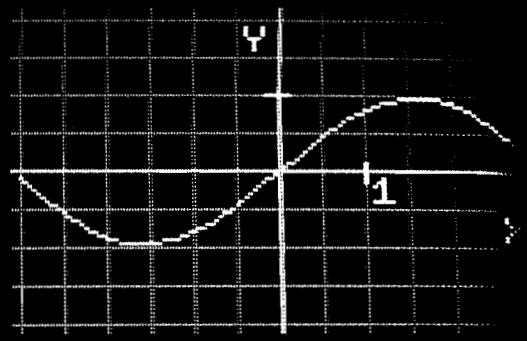


13. Электронная плата ЭВМ «Электроника УКНЦ»

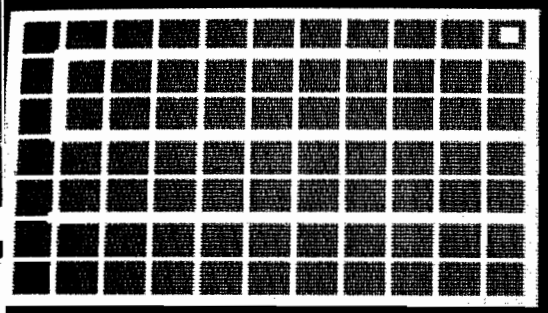


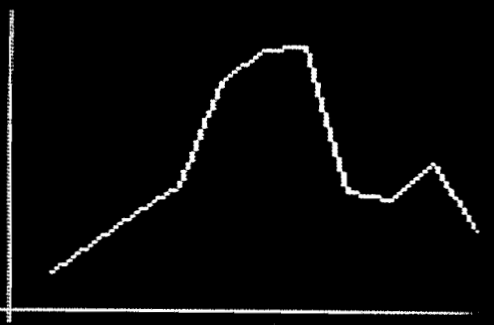
14. ЭВМ «Электроника УКНЦ» со снятой верхней крышкой и клавиатурой

15. Клавиатура ЭВМ «Электроника УКНЦ» и трехслойная полиэтиленовая пленка с контактами (располагается точно под клавишами)




```
___ Е-практикум - 87 ___ (С) МГУ,мехмат,ЛВМ ___ Школа-1 ___
. x:=a
. нц пока x<b-0.1*dx
. . x:=x+dx
. . сместиться_в_точку(x,ф(x))
. кц
кон _____________________________
алг вещ ф ( вещ x )
 дано
 надо
нач
. знач:=sin(x)
```





```
___ Е-практикум - 87 ___ (С) МГУ,мехмат,ЛВМ ___ Школа-1 ___
алг График_радиации
 дано | Восточнее робота - тупик
 надо | построить график изменения радиации в этом тупике
нач вещ a
. сместиться_в_точку(1,радиация)
. a:=1;опустить_перо
. нц пока на_востоке_свободно
. . шаг_на_восток
. . a:=a+1
. . сместиться_в_точку(a,радиация)
. кц
```

```
Р0Й ЦЯЦЯ ЛАРШХ ГЕЛКПИХ НМАВИС,
ТОЧЦА ПЕ Ц ЗУКТУ ФАПЕРОЧ,
ОП УЦАХАКЬ ЛЕЦЯ ФАЛКАЦИС
Н СУГЗЕ ЦИЦУРАКЬЦИЕ МОГ!
[Конец текста]
```

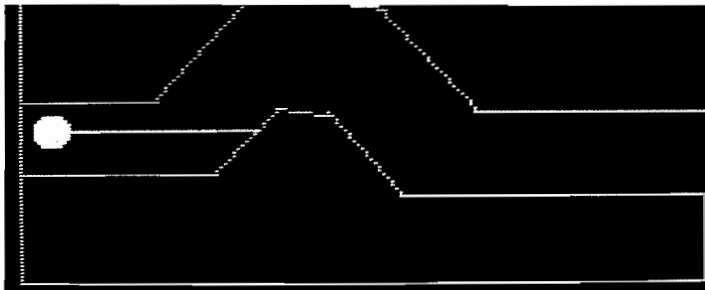
Enter

```
___ Е-87 ___ Школа-1 ___ Март 1988 ___ (С) МГУ,мехмат,ЛВМ ___
. нц пока не в_конце_текста
. . если в_нис(стр,кол(текущ),нов)
. . .то
. . . если нов>10
. . . .то установить_текущ(сд[(нов-10):(нов-10)])
. . . .иначе установить_текущ(сд[нов:нов])
. . . все
. . .иначе
. . . если в_нис(загл,кол(текущ),нов)
. . . .то
. . . . если нов>10
```

Истина
Ложно
Ложно
Истина

16–23. Результаты выполнения алгоритмов, записанных на школьном алгоритмическом языке на экране монитора

16. Алгоритм построения графика функции (исполнитель «Чертежник»)



17. Алгоритм построения графика радиации (в тупике, расположенном справа от Робота (исполнителей «Робот» и «Чертежник»)

```
алг ПРОХОД
 дано | вездеход у западного края коридора
 надо | сместиться к восточному краю
нач
. вездеход
. до_упора
. нц пока можно_вверх
. . налево(90); до_упора
```

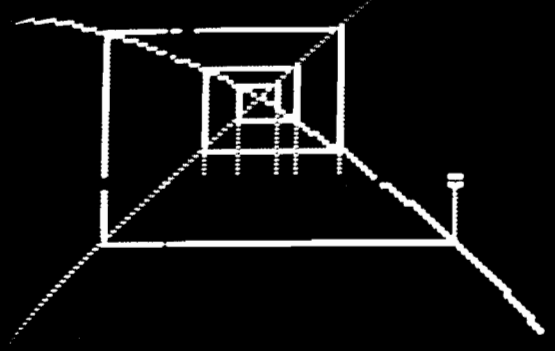
18. Алгоритм шифровки и расшифровки сообщений (исполнитель «Редактор»)

19. Алгоритм движения вперед по извилистой дороге (исполнитель «Вездеход»)

20. Алгоритмы подъема по лестнице (исполнитель «Двуног»)

```
___ Е-87 ___ Школа-1 ___ Март 1988 ___ (С) МГУ,мехмат,ЛВМ ___
кон _____________________________
алг странный_шаг
 дано
 надо
нач
. голову_налево(58)
. полохать(30)
. поднять_правую_ногу(78)
. голову_направо(90)
. полохать(65)
. полохать(15)
```

```
___ E-87 ___ Школа-1 ___ Март 1988 ___ (С) МГУ,мехмат,ЛВМ ___
. . .иначе направо; взять
. . все
. все
. на_север
. нц для м от 1 до у
. . шаг_вперед; хай(3);  кругом; взять; кругом
. кц
. на_запад
. нц для м от 1 до х
. . шаг_вперед;хай(3);    кругом; взять; кругом
. кц
```



Метод итераций

```
___ E-практикум - 87 ___ (С) МГУ,мехмат,ЛВМ ___ Школа-1 ___
. . поз(0,нач);линия(120,нач);поз(113,нач+1);надпись("x")
. . поз(42,0);линия(42,85);поз(44,0);надпись("y")
. . цвет(3);поз(0,84);линия(84,0)
. . цвет(10);поз(0,85-функция(0,к))
. . нц для t от 1 до 24
. . . x:=t*5; линия(x,85-функция(x,к))
. . кц
. . x:=x0;y:=функция(x,к);отметить(x,нач,7)
. . цвет(13);поз(x,нач);линия(x,85-y)
. . нц пока abs(x-y)>5
. . . цвет(11);поз(x,85-y);линия(y,85-y)
```

21. Алгоритм построения пирамиды (исполнитель «Строитель»)

22. Пирамида на поле исполнителя «Строитель» крупным планом

23. Алгоритм нахождения точки пересечения графика функции с прямой $y = x$ методом итераций

24. Синтаксические ошибки при составлении алгоритма мгновенно диагностируются на полях

25. Изображение значений величин на полях помогает обнаружить логические ошибки в алгоритме

```
алг ОСТАТКИ( цел м )
 дано м = 8
 надо
нач цел т, остаток
. нц для т от 1 до м
. . остаток:=mod(2**т)
. кц
кон ____________________
```

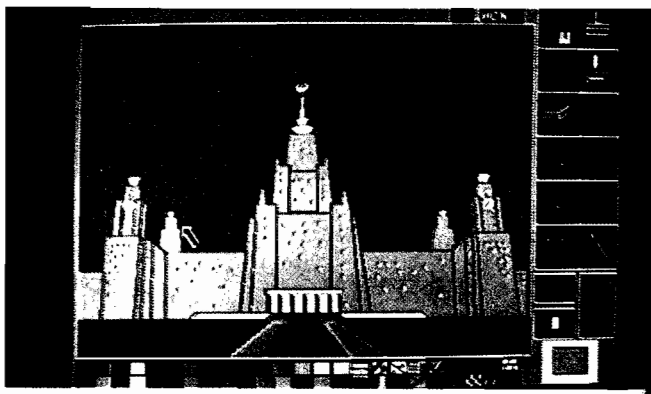
```
Ош. число арг.
```

```
алг Сумма_квадратов( цел N, S )
 дано   N = 100
 надо ! S = 1*1+2*2+...+N*N
нач цел к
. S:=0
. нц для к от 1 до N
. . S:=S+к
. кц
кон ____________________
```

```
0
100
5050
```

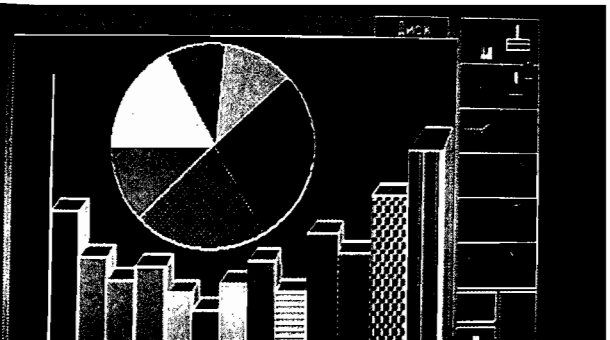
26. Гибкие магнитные диски — средство хранения информации в современных ПЭВМ

29. Штриховой код товара (прямоугольник из черных и белых полосок) на упаковке компакт-кассеты

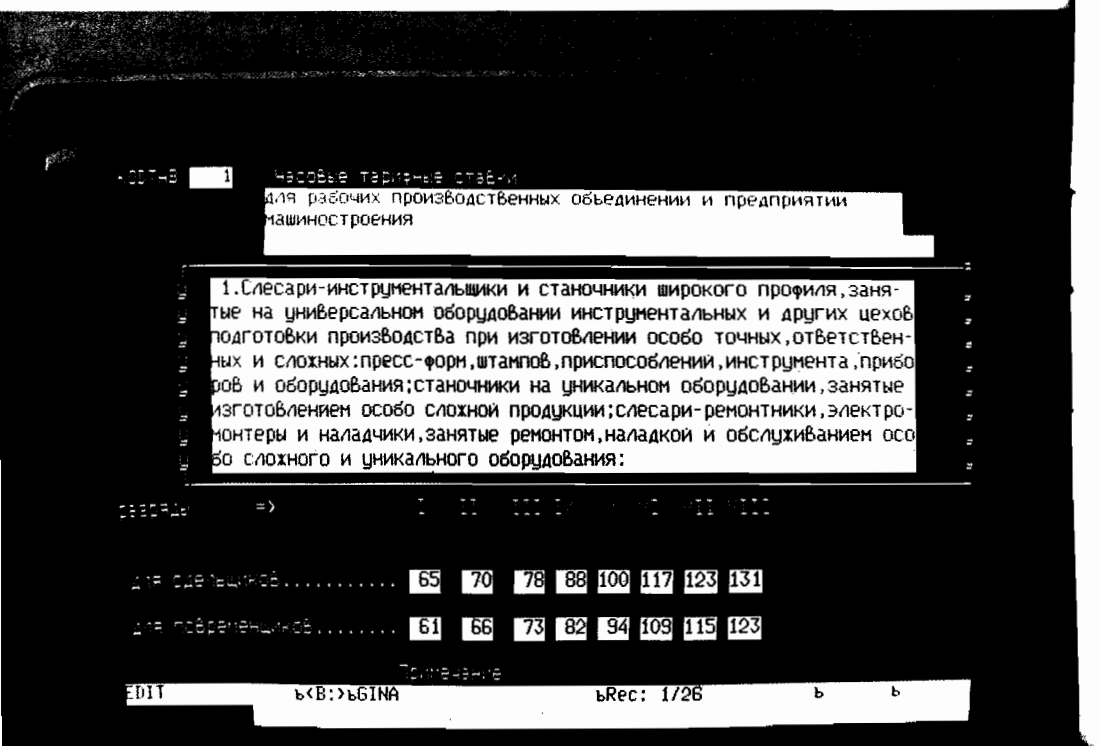


27. Здание МГУ на экране ЭВМ «Корвет» (графический курсор «стрелка» подведен к недокрашенной части рисунка)

28. Устройство ввода «мышь»

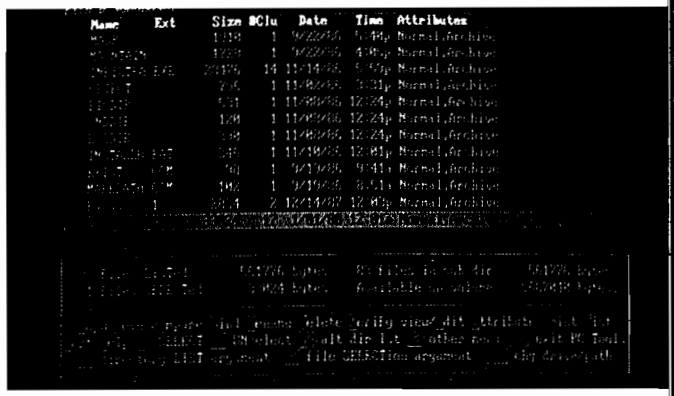


30—31. Программа «Графический редактор», разработанная в НИИ ядерной физики МГУ, помогает рисовать новогодние поздравления (30) и диаграммы (31)



32. Отечественная персональная ЭВМ ЕС-1841

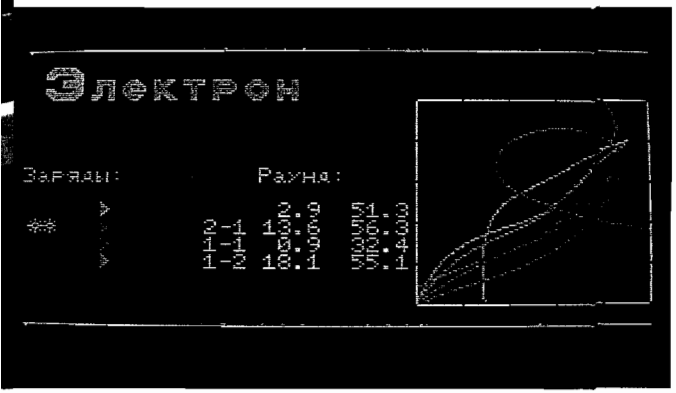
33. Результат обработки экономической информации на экране ПЭВМ ЕС-1841

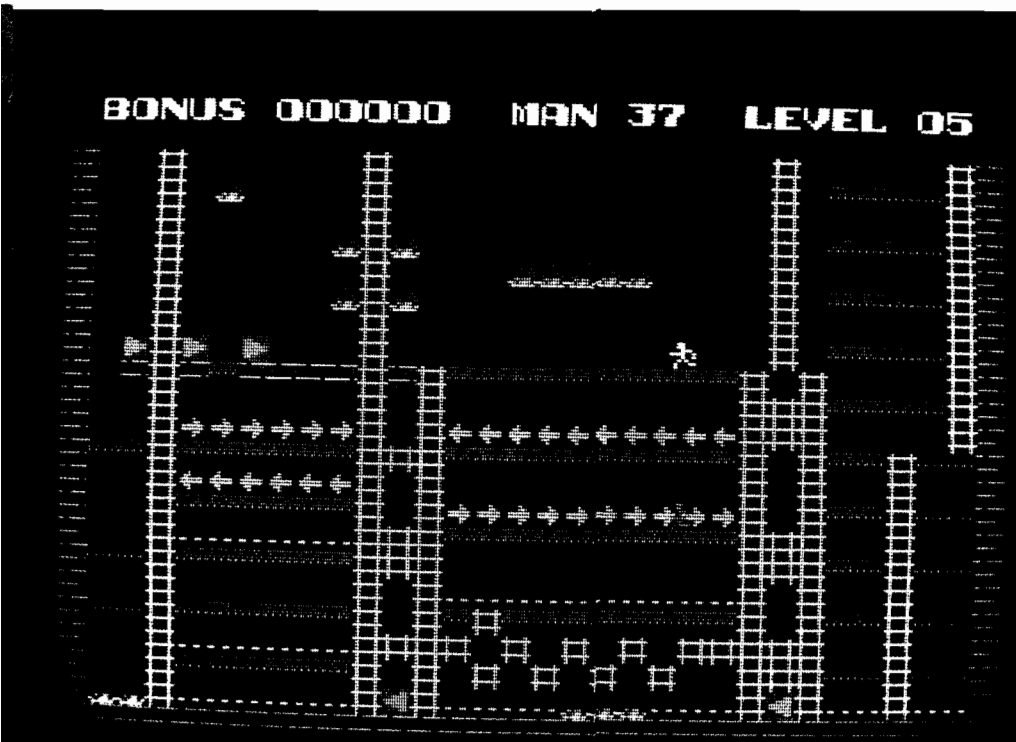


34. Персональная ЭВМ IBM PC и изображение клавиш пианино на ее экране при работе программы «Музыкальный редактор», которая позволяет вводить и проигрывать простейшие мелодии

35. Работа с каталогом информации на магнитном диске

36. Демонстрация работы текстово-графического пакета «Гратекс» на экране ЭВМ «Ямаха»





37—40. Игры на ЭВМ: обучение и досуг

Монитор ЭВМ "Корвет" позволяет формировать изображение в виде мозаики размером 512 точек по горизонтали и 256 точек по вертикали, а монитор ЭВМ "Электроника УКНЦ" — в виде мозаики 640 на 288 точек. Каждая точка может иметь 8 уровней яркости (или 8 цветов): от 0 (точка вообще не светится) до 7 (максимально возможная яркость). Комбинируя светлые и темные точки, можно изображать не только картинки, но и буквы, цифры и другие символы. В ЭВМ "Корвет" символы размещаются в прямоугольниках шириной 8 и высотой 16 точек. Поэтому на экране помещается 256/16=16 строк по 512/8=64 символа. В УКНЦ на экране 24 строки по 80 символов.

19.3. ДИСКОВОД

В школьной ЭВМ используется дисковод (фото 12 вклейки), рассчитанный на гибкие магнитные диски диаметром $5\frac{1}{4}$-дюйма ($\approx$133 мм; название "дюйм" происходит от голландского слова, означающего "большой палец"; 1 дюйм равен 25.4 мм).

Гибкий магнитный диск (***дискета***) представляет собой круг из пластика, покрытый с обеих сторон магнитным слоем. Этот слой похож на покрытие магнитной ленты для обыкновенных магнитофонов. В дисководе, как и в магнитофоне, запись делается с помощью магнитной головки. Только записывается информация не вдоль ленты, а по невидимым кольцевым дорожкам на вращающемся диске. В отличие от магнитной ленты, которую нередко приходится подолгу перематывать, в дисководе магнитную головку можно сразу установить на нужную дорожку. Именно этим объясняется использование дисков, а не лент. Гибкие диски, как и кассеты в магнитофоне, являются сменными — в дисковод можно вставлять разные диски, а один и тот же диск можно использовать в разных дисководах на разных ЭВМ.

19.4. ПРИНТЕР

Использованный в школьной ЭВМ принтер (фото 9 вклейки) относится к так называемым "точечно-матричным" принтерам. Главная деталь таких принтеров — печатающая головка с тонкими иголочками (фото 10 вклейки). Каждую иголочку выдвигает вперед свой электромагнит; когда по его катушке проходит импульс тока. В школьной ЭВМ в головке 9 расположенных вертикально, одна над другой, иголочек. Выдвигаясь, иголочки ударяют по красящей ленте и оставляют на бумаге точку. При печати головка мелкими шажками движется слева направо, после каждого шага печатая очередную колонку точек. На рисунке 80 показано, как из таких точек может получиться буква К.

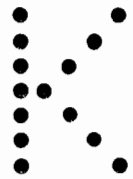


Рис. 80

19.5. ВЗАИМОДЕЙСТВИЕ ОСНОВНЫХ ЧАСТЕЙ ЭВМ. МАГИСТРАЛЬ

Как же связаны между собой процессор, память и устройства ввода/вывода? В современной электронике для этого чаще всего используется ***магистраль***. Магистраль можно представлять как пучок проводов, к которому подключены параллельно все компоненты ЭВМ (рис. 81). Магистраль содержит шину адреса, шину данных и некоторые дополнительные провода (например, для указания того, надо ли данные по некоторому адресу записать или прочесть). Посылая по магистрали электрические сигналы, компоненты ЭВМ могут передавать информацию друг другу.

Рис. 81

Как процессор управляет устройствами? При подключении через магистраль любая команда устройству выглядит как команда на запись или чтение некоторой информации по специальному адресу. В ЭВМ "Электроника УКНЦ", например, адрес может меняться в диапазоне от 0 до 65535. Но только адреса от 0 до 57343 относятся к памяти. Остальные адреса распределены между разными устройствами (часть адресов не задействована вовсе). Так, устройству печати (принтеру) выделены адреса 65356 и 65358, клавиатуре — адреса 65392 и 65394.

Вызов команды принтера выглядит так: процессор посылает по магистрали код адреса 65356, код команды и сигнал "запись". Каждое подключенное к магистрали устройство анализирует код адреса. Принтер обнаруживает "свой" адрес и включает схемы, которые по сигналу "запись" и состоянию шины данных выполняют соответствующую команду. Посылая разную информацию, можно заставить принтер выполнять разные команды. Аналогичным образом, по сигналу "чтение" и коду адреса 65394 клавиатура посылает по шине данных информацию о своем состоянии, проанализировав которую можно узнать, какую клавишу нажал человек. Напомним, что слова "посылает", "анализирует", "обнаруживает" и пр. означают просто срабатывание соответствующих схем и вентилей, появление в проводниках высокого (+5В) или низкого (0В) напряжения.

Таким образом, для управления устройствами в УКНЦ нет никаких специальных команд — используются уже известные нам команды "переслать из x в y", "сравнить x с y" и др. (в которых указываются выделенные устройству адреса). А разные устройства отличаются друг от друга только тем, какие именно адреса им выделены и как (какими наборами 0 и 1) кодируются их команды и состояния.

19.6. УСТРОЙСТВА И ИСПОЛНИТЕЛИ

С логической точки зрения все компоненты ЭВМ, кроме процессора,— это самые обычные исполнители. Память ЭВМ, например,— это исполнитель с двумя командами: прочесть/записать слово по заданному адресу. Лишь процессор в ЭВМ играет особую роль: автоматически выполняя программу, он командует всеми остальными устройствами.

Для управления устройствами не обязательно писать машинные программы. Поскольку устройства отличаются от Робота и Чертежника только системой команд, то для управления ими можно составлять алгоритмы и на алгоритмическом языке. Покажем, как это делается на примере монитора и клавиатуры.

19.7. ИСПОЛНИТЕЛЬ МОНИТОР (ЭКРАН)

С логической точки зрения удобно считать, что монитор — это исполнитель, который может окрасить в любой цвет любую точку экрана, а кроме того, умеет устанавливать курсор в заданную позицию экрана и выводить в эту позицию символ. На алгоритмическом языке эти команды записываются в виде:

цвет (**арг** **цел** n)	выбор текущего цвета или яркости от 0 (черный цвет) до 7 (белый)
точка (**арг** **цел** i, j)	окраска точки (i, j) в текущий цвет
поз (**арг** **цел** y, x)	установка курсора в позицию x строки y
вывсим (**арг** **сим** s)	вывод символа s в позицию курсора (курсор смещается на символ вправо)

При выполнении команды **вывод** ЭВМ преобразует тексты и значения величин в строку символов и вызывает соответствующую последовательность команд "вывсим". Встретив служебное слово **нс**, ЭВМ вызывает команду "поз", указывая в ней координаты начала следующей строки, и т. п.

19.8. ИСПОЛНИТЕЛЬ КЛАВИАТУРА

Аналогично, при выполнении команды **ввод** ЭВМ последовательно запрашивает у клавиатуры коды клавиш до тех пор, пока человек не нажмет специальную клавишу перехода в начало следующей строки (на ней обычно изображена изогнутая стрелка). В процессе ввода ЭВМ вызывает команды "вывсим" для отображения вводимых символов на экране. По окончании ввода ЭВМ

анализирует полученную строку символов и присваивает указанным в команде **ввод** величинам соответствующие значения.

В алгоритмическом языке можно и явно запросить у клавиатуры код нажатой клавиши с помощью команды

клав (**рез** **цел** k).

При выполнении вызова "клав(k)" ЭВМ запрашивает у клавиатуры код клавиши (либо ждет, пока человек нажмет какую-нибудь клавишу, если она еще не нажата). Код нажатой клавиши (целое число от 0 до 255) присваивается величине k. Изображение на экране монитора при этом не меняется.

УПРАЖНЕНИЯ

1. Для подключения Робота к УКНЦ выделены два адреса: 65352 и 65354. Придумайте способ кодировки команд Робота и передаваемой им информации. Используя эту кодировку, переведите в машинную программу алгоритмы: А1; А31; А38; А40.

2. Опишите, как будет выполняться следующий алгоритм:

```
алг квадраты                                          (А90)
нач цел n, k
| n:=0; k:=0
| нц пока k≠13   |13 — код клавиши перехода в начало сле-
| | n:=n+1       |дующей строки
| | поз (1,1); вывод "n=", n, ", n*n=", n*n
| | клав (k)
| кц
кон
```

3. Составьте алгоритм, который очищает (заполняет пробелами) весь экран и устанавливает курсор в его левый верхний угол.

4. Составьте алгоритм, который выводит строку символов **лит** t на экран вертикально: сверху вниз, начиная с позиции курсора.

5. Сопоставим первым пяти командам Робота по одной букве:

Команда	вверх	вниз	влево	вправо	закрасить
Буква	В	Н	Л	П	К

По образцу упражнения 2 составьте алгоритм, который при нажатии на одну из клавиш "В", "Н", "Л", "П", "К" выдает соответствующую команду Роботу, потом ждет нажатия на следующую клавишу, опять выдает команду Роботу и т. д. до тех пор, пока не будет нажата клавиша перехода в начало следующей строки.

6. Аналогично упражнению 5 составьте алгоритм, который при нажатии на клавиши "В", "Н", "Л", "П" рисует с помощью Чертежника отрезок единичной длины в соответствующем направлении.

7. Составьте алгоритм, который с помощью команды "точка" рисует
а) горизонтальный;
б) вертикальный;
в) диагональный;
г*) произвольный
отрезок с заданными концами.

8. Составьте алгоритм, который с помощью команды "точка" рисует: а) прямоугольник; б) окружность; в) круг.

9. Решите упражнение 6, используя вместо Чертежника вспомогательные алгоритмы из упражнений 7а и 7б (решение этой задачи можно назвать простейшим графическим редактором).

§ 20. РАБОТА ЭВМ В ЦЕЛОМ

Устройство и работу ЭВМ можно изучать на разных уровнях. Можно интересоваться физическими основами электронной обработки информации (§ 17). Можно, забыв про напряжение, транзисторы, вентили, микросхемы и пр., изучать логическое устройство процессора: систему его команд, алгоритм работы, структуру хранимой во внутренней памяти информации (§ 18). В этом параграфе мы поднимемся еще на один уровень и попробуем описать работу ЭВМ в целом.

20.1. АЛГОРИТМИЧЕСКИЙ ЯЗЫК И МАШИННЫЕ КОДЫ

ЭВМ может выполнять программы, записанные на разных языках программирования: на Бейсике, Паскале, школьном алгоритмическом языке и др. В § 18 мы познакомились с программами в машинных кодах и теперь знаем, что фактически ЭВМ всегда работает по такой — ***машинной*** — программе. Программа на алгоритмическом языке, однако, и программа в машинных кодах — это, как говорится, "две большие разницы". Возникает вопрос: как же ЭВМ выполняет программы, записанные на алгоритмическом или ином языке программирования?

Существует два принципиально разных способа выполнения таких программ. Они называются компиляцией и интерпретацией.

20.2. КОМПИЛЯЦИЯ

Программа на алгоритмическом языке — это информация, а ЭВМ — универсальная машина для обработки информации. Поэтому можно составить машинную программу (она называется ***компилятором***), при выполнении которой ЭВМ обработает ***текст***

программы на алгоритмическом языке и преобразует его в эквивалентную программу в машинных кодах (т. е. в последовательность байтов). Само такое преобразование называется ***компиляцией***. Поскольку в результате компиляции получается программа в машинных кодах, то ее уже можно поместить в память ЭВМ и выполнить.

Таким образом, при компиляции работа происходит в два этапа: на первом этапе ЭВМ переводит программу с алгоритмического языка в машинные коды, а на втором — выполняет полученную программу в машинных кодах.

20.3. ИНТЕРПРЕТАЦИЯ

Можно, однако, составить другую программу в машинных кодах (она называется ***интерпретатором***), при выполнении которой ЭВМ будет анализировать ***текст*** программы на алгоритмическом языке и сразу выполнять предписанные этой программой действия, не переводя ее в машинные коды. Такой способ выполнения программы называется ***интерпретацией***.

20.4. КОМПИЛЯЦИЯ И ИНТЕРПРЕТАЦИЯ

Разницу между компиляцией и интерпретацией можно пояснить с помощью аналогии. Пусть требуется приготовить пирог по рецепту, написанному на иностранном языке. Есть два пути. Можно сначала перевести (скомпилировать) рецепт на родной язык и лишь затем готовить пирог по рецепту на родном языке. Можно, однако, по мере чтения и перевода рецепта сразу выполнять требуемые им действия (это соответствует интерпретации). В последнем случае мы не получим текста рецепта на родном языке, а сразу получим пирог. Правда, если пирог придется готовить несколько раз, рецепт придется переводить многократно.

20.5. ПРОГРАММА НАЧАЛЬНОЙ ЗАГРУЗКИ

Для того чтобы можно было выполнить алгоритм, надо поместить в память ЭВМ и начать выполнять компилятор или интерпретатор алгоритмического языка. Сам компилятор — это программа в машинных кодах, т. е. последовательность 0 и 1 (информация!). Конечно, процессор может прочесть эту информацию с магнитного диска, но при этом он должен работать по какой-то другой (еще одной!) программе — ведь единственное, что он умеет, это выполнять машинные программы. Эту "еще одну" программу тоже можно прочесть с диска или ввести с какого-нибудь из устройств ввода. Но кто это может сделать? Только процессор, а ему понадобится новая программа и т. д. Получается какой-то порочный круг!

Чтобы разобраться в этом вопросе, рассмотрим, что происходит при включении ЭВМ. Сразу после включения процессор начинает выполнять специальную ***программу начальной загрузки***. Эта программа записывается в постоянную память (ПЗУ) при изготовлении ЭВМ на заводе и не исчезает при выключении ЭВМ. Эту программу читать с диска не надо.

Что же это за программа? На машине учителя (как и на большинстве персональных ЭВМ) выполнение программы начальной загрузки состоит в том, что процессор читает с определенного, заранее фиксированного места на диске другую машинную программу (она называется ***операционной системой***), записывает ее в память и помещает в СК адрес первой команды этой программы (т. е. начинает ее выполнять). На машине ученика дисков нет и программа начальной загрузки либо ждет и получает программу с машины учителя (в классе "Электроника УКНЦ"), либо запускает другую программу из ПЗУ, например интерпретатор языка Бейсик ("Ямаха", "Корвет").

20.6. ОПЕРАЦИОННАЯ СИСТЕМА (ОС)

Итак, в результате начальной загрузки в ЭВМ обычно "загружается" программа, называемая операционной системой. В отличие от программы начальной загрузки операционная система читается с диска и поэтому может быть легко изменена — достаточно записать на диск новую информацию.

Что же делает операционная система? Прежде всего она позволяет человеку выбрать, с какой именно программой он желает дальше работать, т. е. запрашивает у человека имя программы, помещает ее в память и начинает выполнять. Человек может выбрать, например редактор текстов, компилятор с языка Паскаль или систему программирования КуМир для школьного алгоритмического языка. Дальше ЭВМ работает уже по новой программе. Система КуМир, например, позволяет человеку вводить и выполнять алгоритмы на школьном алгоритмическом языке (результаты изображаются на экране монитора — фото 16—23 вклейки). Помните: "Дедка за репку, бабка за дедку ... мышка за кошку — вытянули репку"? Так и у нас: чтобы выполнить алгоритм ("вытянуть репку"), начинать приходится с программы начальной загрузки ("мышки").

Кроме загрузки и выполнения других программ, операционная система обычно позволяет переписывать информацию с одного диска на другой, распечатывать ее, стирать ненужную информацию, узнавать, сколько на диске незанятого места, и т. п. Все эти действия выполняются по командам человека (при нажатии соответствующих клавиш).

УПРАЖНЕНИЯ

1. Какая информация хранится в памяти ЭВМ при интерпретации алгоритма на алгоритмическом языке?

2. Опишите результат выполнения алгоритма

```
алг интерпретатор (арг лит t)                    (А91)
 дано | строка t состоит только из букв "В", "Н", "Л", "П", "К"
 надо |Робот выполнил соответствующую последовательность
      |команд (соответствие задано в упражнении 5 § 19)
нач цел i
| нц для i от 1 до длин (t)
| | выбор
| | | при t[i] ="В": вверх
| | | при t [i] ="Н": вниз
| | | при t [i] ="Л": влево
| | | при t [i] ="П": вправо
| | | при t [i] ="К": закрасить
| | все
| кц
кон
```

при вызове с аргументом "КВВКНЛКППК". Объясните, почему алгоритм называется интерпретатором.

3. Составьте алгоритм:

```
алг компилятор (арг лит t)
 дано |строка t состоит только из букв "В", "Н", "Л", "П", "К"
 надо |на экран выведена машинная программа (столбик слов,
      |где каждое слово изображено последовательностью 0 и 1),
      |которая содержит вызовы соответствующих команд Робо-
      |та (используйте ваше решение упражнения 1 § 19)
```

Объясните, почему алгоритм называется компилятором.

4*. Составьте алгоритм:

```
алг компилятор (арг лит t)
 дано |строка t состоит только из букв "В", "Н", "Л", "П", "К"
 надо |на экран выведен текст алгоритма на алгоритмическом
      |языке, содержащий соответствующую последовательность
      |вызовов команд Робота
```

Объясните, почему алгоритм называется компилятором.

5. На некоторых ЭВМ операционная система размещена не на диске, а в ПЗУ. Опишите достоинства и недостатки такого подхода.

ГЛАВА 3. ПРИМЕНЕНИЯ ЭВМ

Появление и распространение ЭВМ все сильнее меняет окружающий нас мир. В этой главе мы расскажем о некоторых применениях ЭВМ. Разумеется, нет никакой возможности сколько-нибудь подробно описать в учебнике реальные применения ЭВМ. Мы выйдем из положения так: описав общий характер применений ЭВМ в той или иной области, мы проиллюстрируем отдельные методы обработки информации в этой области на простейших примерах (т. е., рассказывая об океанских лайнерах, продемонстрируем корыто и объясним, почему оно плавает).

Но вначале познакомимся с понятием информационной модели и моделированием действительности на ЭВМ.

§ 21. КОДИРОВАНИЕ ИНФОРМАЦИИ ВЕЛИЧИНАМИ АЛГОРИТМИЧЕСКОГО ЯЗЫКА. ИНФОРМАЦИОННЫЕ МОДЕЛИ

21.1. ПОНЯТИЕ ИНФОРМАЦИОННОЙ МОДЕЛИ

Обрабатываемая на ЭВМ информация должна быть представлена в виде значений величин используемого языка программиро-

вания. Мы используем школьный алгоритмический язык и потому будем кодировать информацию с помощью величин этого языка.

Набор величин, содержащий всю необходимую информацию об исследуемых объектах и процессах, в информатике называется ***информационной моделью***. Как и любая модель, информационная модель содержит не всю информацию о моделируемых явлениях, а только ту ее часть, которая нужна для рассматриваемых задач.

21.2. ПРОСТЕЙШИЙ ПРИМЕР ИНФОРМАЦИОННОЙ МОДЕЛИ

Пусть в кинозале 28 рядов по 20 мест, а цена билета зависит только от номера ряда. Закодируем эту информацию:

```
лог таб продано [1:20, 1:28] |продано [i, j] =да ↔          (M1)
                             |     место i ряда j продано
цел таб цена [1:28]          |цена [j] =цена билета в j-м ряду
```

Используя эту информационную модель, легко подсчитать общее число n проданных билетов:

```
n:=0
нц для j от 1 до 28          |для каждого ряда
| нц для i от 1 до 20        |для каждого места в ряду
| | если продано [i, j]      |если место продано
| | | то n:=n+1              |то учесть его
| | все
| кц
кц
```

или выручку s от продажи билетов:

```
s:=0
нц для j от 1 до 28          |для каждого ряда
| нц для i от 1 до 20        |для каждого места в ряду
| | если продано [i, j]      |если место продано
| | | то s:=s+цена [j]       |то учесть его цену
| | все
| кц
кц
```

Описанная модель кинозала весьма условна — она рассчитана только на один сеанс, не учитывает возможности бронирования мест, продажи абонементов и отражает лишь те свойства реального мира, которые важны для продажи билетов и подсчета выручки. В модели не учитывается, например, количество входов и выходов из кинозала, наличие или отсутствие проходов в зале и пр.

21.3. ИНФОРМАЦИОННАЯ МОДЕЛЬ ТРАНСПОРТНОЙ СЕТИ

Пусть в некоторой стране 100 городов и между некоторыми из них летают самолеты. Авиатрассы проложены так, что из любого города можно перелететь в любой другой (возможно с пересадками).

Построим информационную модель, используя которую можно отвечать на вопросы, как перелететь из одного города в другой, каково минимальное число пересадок и какова минимальная длина пути при таком полете: (M2)

```
лит таб назв [1:100]          |назв [i] — название i-го города
лог таб прям [1:100, 1:100]   |прям [i, j] =да↔между городами
                              |   i и j есть прямой авиарейс
вещ таб расст [1:100, 1:100]  |расст [i, j] =расстояние между
                              |   городами i и j
```

В таблице "расст" в этой модели хранятся расстояния только между такими городами i и j, для которых прям [i, j] =да.

Используя модель M2, можно, например, заставить ЭВМ проанализировать, как добраться из города i1 в город i2, совершив не более одной пересадки (ниже используются промежуточные величины цел p, лог найден пункт пересадки):

```
вывод "Из города", назв [i1], "в город", назв [i2], нс
если прям [i1, i2] то вывод "есть прямой рейс"
| иначе
| p:=0; найден пункт пересадки :=нет
| нц пока p<100 и не найден пункт пересадки
| | p:=p+1
| | найден пункт пересадки :=прям [i1, p] и прям [p, i2]
| кц
| если найден пункт пересадки
| | то вывод "есть маршрут с одной пересадкой в", назв[p]
| | иначе вывод "с одной пересадкой добраться нельзя"
| все
все
```

21.4. КОДИРОВАНИЕ ГЕОМЕТРИЧЕСКОЙ ИНФОРМАЦИИ

Основы числового кодирования геометрической информации заложил великий французский ученый XVII в. Рене Декарт: любую точку плоскости можно задать парой чисел — ее ***декартовыми*** координатами. А от числового кодирования точек нетрудно перейти к кодированию более сложных геометрических объектов, например фигур школьной планиметрии (рис. 82):

а) Точка	вещ x, y	координаты точки	(M3)
б) Окружность	вещ r вещ a, b	радиус окружности координаты центра	(M4)

в) Прямая	**вещ** x1, y1 **вещ** x2, y2	координаты двух точек на прямой (М5)
г) Луч	**вещ** x1, y1 **вещ** x2, y2	координаты начала луча координаты точки на луче (М6)
д) Отрезок	**вещ** x1, y1 **вещ** x2, y2	координаты начала отрезка координаты конца отрезка (М7)
е) Треугольник	**вещ** x1, y1 **вещ** x2, y2 **вещ** x3, y3	координаты вершин треугольника (М8)
ж) n-угольник	**вещ таб** x [1:n] **вещ таб** y [1:n]	(x [i], y [i]) — координаты i-й вершины (М9)
з) Прямоугольник со сторонами, параллельными осям *Ox*, *Oy*	**вещ** x min **вещ** x max **вещ** y min **вещ** y max	Точка (x, y) принадлежит прямоугольнику ↔ x min ⩽ x ⩽ x max y min ⩽ y ⩽ y max (М10)
и) Квадрат	**вещ** a, b **вещ** u, v	центр квадрата координаты вектора из центра к одной из вершин квадрата (М11)

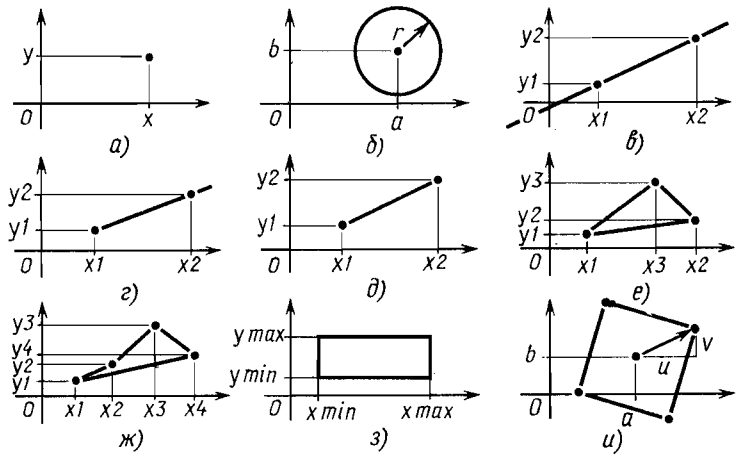


Рис. 82

Пользуясь этими моделями, можно переводить геометрические понятия на алгоритмический язык:

Геометрия	Алгоритмический язык
Длина отрезка (модель М3)	sqrt ((x1 — x2) **2 + (y1 — y2) **2)
Точка М3 внутри окружности М4	(x — a) ** 2 + (y — b) ** 2 < r **2
Квадрат длины диагонали n-угольника М9	(x [i] — x [j]) ** 2 + (y [i] — y [j]) ** 2
Периметр n-угольника М9	p:=0 **нц для** i **от** 1 **до** n — 1 | p:=p+длина (i, i+1) **кц** p:=p+длина (n, 1)
Окружность М4 внутри прямоугольника М10	x min < a — r **и** a + r < x max **и** y min < b — r **и** b + r < y max
Площадь прямоугольника М10	(x max — x min) * (y max — y min)
Площадь квадрата М11	2 * (u ** 2 + v ** 2)

Один и тот же геометрический объект можно кодировать по-разному. Например, отрезок можно задать и так:

вещ a, b \|координаты середины отрезка (М12)
вещ l \|длина отрезка
вещ φ \|угол наклона отрезка к оси *Ox*

Наконец, даже одни и те же числа можно представить разными величинами алгоритмического языка, например для задания точки можно использовать таблицу

вещ таб x [1:2] \|точка с координатами (x [1], x [2]) (М13)

21.5. ИНФОРМАЦИОННАЯ МОДЕЛЬ ОБСТАНОВКИ НА ПОЛЕ РОБОТА

Вспомните алгоритм А62 (п. 14.5) — в нем была построена информационная модель радиационной обстановки в горизонтальном коридоре длины n на поле Робота:

вещ таб a [1:n] \| a [i] = уровень радиации в клетке i (М14)

Выбор информационной модели зависит от того, какие задачи мы собираемся решать с ее помощью. Модель М14, например, не

позволяет узнать количество закрашенных клеток или боковых выходов из коридора. Мы можем составить более полную информационную модель коридора, включив в нее информацию о положении Робота, стенах, закрашенных клетках и температуре:

```
цел x |положение Робота:                                   (М 15)
      |  x=1 — Робот в левой клетке коридора
      |  x=n — Робот в правой клетке коридора
лог таб выше [1:n] |выше [i]=да ↔ выше i-й клетки стена
лог таб ниже [1:n] |ниже [i]=да ↔ ниже i-й клетки стена
лог таб закр [1:n] |закр [i]=да ↔ i-я клетка закрашена
вещ таб a[1:n]     |a [i]=уровень радиации в клетке i
вещ таб t[1:n]     |t [i]=температура      в клетке i
```

Аналогично можно построить информационную модель поля Робота, точнее говоря, прямоугольника на этом поле, скажем, размером 32 клетки в ширину и 16 в высоту (столбцы прямоугольника занумеруем слева направо от i=1 до i=32, а ряды — сверху вниз от j=1 до j=16):

```
                                                          (М16)
цел x |1≤x≤32, x — координата Робота (номер столбца)
цел y |1≤y≤16, y — координата Робота (номер строки)
лог таб  ниже [1:32,1:16] |ниже [i, j] =да ↔ ниже стена
лог таб  выше [1:32,1:16] |выше [i, j] =да ↔ выше стена
лог таб левее [1:32,1:16] |левее [i, j] =да ↔ левее стена
лог таб правее [1:32,1:16] |правее [i, j] =да ↔ правее стена
лог таб  закр [1:32,1:16] |закр [i, j] =да ↔ клетка закрашена
вещ таб a [1:32,1:16]     |a [i, j] — уровень радиации
вещ таб t [1:32,1:16]     |t [i, j] — температура в клетке
```

Пользуясь моделью М16, можно имитировать любые действия по управлению Роботом в прямоугольнике размером 32 на 16:

Управление Роботом	Имитация управления Роботом
вправо	x:=x+1
вверх	y:=y−1
закрасить	закр [x, y]:=да

нц n **раз**
  если температура>100
  то s:=s+радиация
  все
  вниз
кц
	нц n **раз**
	  если t [x, y]>100
	  то s:=s+a [x, y]
	  все
	  y:=y+1
	кц

если сверху стена **то** влево **все**	**если** выше [x, y] =**да** **то** x:=x−1 **все**
утв | Робот в левом верхнем углу	**утв** x=1 **и** y=1
утв | Робот в правом нижнем углу	**утв** x=32 **и** y=16

21.6. КОДИРОВАНИЕ АЛГОРИТМОВ УПРАВЛЕНИЯ ИСПОЛНИТЕЛЯМИ

Вспомните соответствие между первыми 5 командами Робота и буквами "В", "Н", "Л", "П", "К" из упражнения 5 § 19. При таком соответствии алгоритм управления Роботом, содержащий только эти команды, можно закодировать одной литерной величиной:

```
лит t |алгоритм для Робота из букв В, Н, Л, П, К          (М17)
```

Именно эта информационная модель была использована в упражнениях 2—4 предыдущего параграфа.

Рассмотрим еще один пример. Закодируем каждую из команд Чертежника одним или тремя вещественными числами:

Команда	Код
поднять перо	1
опустить перо	2
сместиться в точку (x, y)	3 x y
сместиться на вектор (a, b)	4 a b

Тогда всякую последовательность команд Чертежника можно задать линейной таблицей

```
вещ таб a [1:n] |последовательность кодов и аргументов  (М18)
                |команд Чертежника
```

Например, последовательность команд

```
поднять перо                      |1
сместиться в точку (0, 0)         |3 0 0
опустить перо                     |2
сместиться на вектор (1.5, −1.5)  |4 1.5 −1.5
поднять перо                      |1
```

можно задать таблицей **вещ таб** a [1:9]:

вещ таб a

1	3	0	0	2	4	1.5	−1.5	1
1	2	3	4	5	6	7	8	9

Как узнать, какую последовательность команд Чертежника кодирует эта таблица? В а[1] записано число 1 — код команды "поднять перо". Поскольку эта команда аргументов не имеет, то следующая команда начинается в а[2]. а[2]=3 (команда "сместиться в точку"). Эта команда имеет два аргумента, значит, а[3] и а[4] — ее аргументы, следующая команда начинается в а[5] и т. д.

Выполнить закодированную в таблице а последовательность команд можно с помощью следующего алгоритма:

```
алг интерпретатор (арг цел n, вещ таб a[1:n])                (A92)
 дано |таблица a содержит закодированную последовательность
      |команд Чертежника (модель M18)
 надо |Чертежник выполнил эту последовательность команд —
      |нарисована соответствующая картинка
нач цел i
| i:=1
| нц пока i⩽n
| | выбор
| | | при a[i]=1 : поднять перо; i:=i+1
| | | при a[i]=2 : опустить перо; i:=i+1
| | | при a[i]=3 : сместиться в точку (a[i+1], a[i+2]); i:=i+3
| | | при a[i]=4 : сместиться на вектор (a[i+1], a[i+2]); i:=i+3
| | все
| кц
кон
```

Величина i в этом алгоритме играет роль Счетчика Команд, а таблица а — роль машинной программы (части памяти ЭВМ).

УПРАЖНЕНИЯ

1. В рамках модели M1 запишите на алгоритмическом языке: а) начало продажи билетов на новый сеанс (приведение модели в состояние, когда все места свободны); б) продажу билета; в) поиск двух соседних свободных мест; г) поиск трех соседних свободных мест.

2. Измените модель M1, считая, что места в центре зала стоят дороже, чем места по краям. Запишите на алгоритмическом языке подсчет выручки в измененной модели.

3. Измените модель M1, считая, что места могут бронироваться. Запишите на алгоритмическом языке бронирование и разбронирование мест.

4. Измените модель M1, считая, что в зале проходит 6 сеансов в день — 4 по одной цене и 2 по другой, а билеты продают только "на завтра".

5. В рамках модели M1 запишите на алгоритмическом языке подсчет выручки, считая, что место номер 13 в 13-м ряду продается со скидкой 75%, а все остальные места в 13-м ряду, а также все 13-е места в других рядах продаются со скидкой 50%.

6. Составьте информационную модель кинозала, в котором ряды имеют разное количество мест.

7. В рамках модели M2 запишите на алгоритмическом языке фрагмент алгоритма, который выводит на экран:

а) все маршруты из города i1 в город i2 ровно с одной пересадкой и длину каждого из этих маршрутов;

б) все маршруты из i1 в i2 с одной пересадкой в порядке возрастания длины маршрута;

в) кратчайший маршрут из i1 в i2 с одной пересадкой;

г) сообщение о том, можно ли добраться из i1 в i2 с двумя пересадками.

8. Дано, что линейная таблица **цел таб** p[1:m] содержит номера всех городов, в которые можно добраться из i1, делая не более двух пересадок. В рамках модели M2 запишите фрагмент алгоритма, который выводит на экран сообщение о том, можно ли добраться из i1 в i2, делая не более трех пересадок.

9. Составьте информационную модель линий и станций Московского метро, учитывающую время в пути между соседними станциями и примерное время, которое тратится на пересадки. Переформулируйте упражнение 7 для этой модели и решите его.

10. Закодируйте величинами алгоритмического языка: а) угол; б) пару параллельных прямых; в) произвольно расположенный прямоугольник (все на плоскости).

11. Составьте алгоритм "**алг вещ** S (**арг вещ** x1, y1, x2, y2, x3, y3)", вычисляющий площадь треугольника по формуле Герона. Используя его как вспомогательный, напишите фрагменты алгоритмов, которые вычисляют:

а) высоту треугольника M8, опущенную из данной вершины;

б) расстояние от точки M3 до прямой M5;

в) касается ли окружность M4 прямой M5.

12. Напишите формулы перехода от модели отрезка M7 к модели M12 и обратно.

13. Напишите формулы, выражающие координаты всех 4 вершин квадрата через величины модели M11.

14. Запишите на алгоритмическом языке условие "отрезок M7 внутри окружности M4".

15. Даны два прямоугольника со сторонами, параллельными осям координат (модель M10): xmin1, xmax1, ymin1, ymax1 и xmin2, xmax2, ymin2, ymax2. Запишите фрагмент алгоритма, который выводит на экран: а) площадь пересечения; б) периметр объединения этих прямоугольников.

16. Закодируйте величинами алгоритмического языка следующие геометрические объекты в пространстве: а) точка; б) сфера; в) прямая; г) плоскость; д) треугольная пирамида;

е) параллелепипед; ж) прямоугольный параллелепипед с ребрами, параллельными осям координат.

17. Информация о расположении стен, хранящаяся в таблицах "выше", "ниже", "правее" и "левее" модели М16, избыточна. Например, если правее [1, 1]=**да** (т. е. между 1-й и 2-й клетками первого ряда стена), то левее [2, 1] тоже обязательно равно **да**. Измените модель М16 так, чтобы информация о стенах кодировалась более экономно, без дублирования.

18. Алгоритм управления Роботом закодирован строкой **лит** t (модель М17), в которой ровно две буквы К. Составьте алгоритм, определяющий, сколько клеток (одну или две) закрасит Робот при выполнении соответствующего алгоритма.

19. Алгоритм управления Роботом закодирован строкой **лит** t (модель М17). Составьте алгоритм, который из **арг лит** t делает строку **рез лит** k, заменяя:

а) каждую группу подряд идущих букв К на одну букву К;

б) каждую группу от 2 до 9 повторяющихся букв В, Н, П, Л на цифру от 2 до 9, за которой следует повторяющаяся буква (при t="ВВНПППППППППППЛ" должна получиться строка k="2ВН9П2ПЛ");

в) каждую группу повторяющихся букв В, Н, П, Л на число повторений, за которым следует повторяющаяся буква (при t="ВВНПППППППППППЛ" должна получиться строка k="2ВН11ПЛ").

20. Составьте алгоритм "**алг** интерпретатор (**арг лит** k)" для выполнения программ, полученных в упражнениях 19 б и 19 в.

21. В строке **лит** t встречаются только буквы В, Н, П, Л, К, Х, а в строке **лит** x — только буквы "В", "Н", "П", "Л", "К". Составьте алгоритм, который:

а) выполняет строку t, считая, что буква Х кодирует команду вызова вспомогательного алгоритма, закодированного в строке x;

б) из строки t делает строку k, подставляя вместо каждой буквы Х содержимое строки x.

22. Измените модель М17 так, чтобы она была применима к алгоритмам управления Роботом, в которых, кроме команд "вверх", "вниз", "вправо", "влево", "закрасить", встречаются логические команды Робота ("сверху стена", "клетка закрашена" и др.), а также цикл **пока**. Составьте алгоритм-интерпретатор для этой модели.

23. Рассмотрим модель М18 как информационную модель некоторой картинки Х, состоящей только из отрезков. По образцу алгоритма А92 составьте алгоритм, который с помощью Чертежника рисует картинку Y:

а) симметричную Х относительно оси Ox;

б) симметричную Х относительно оси Oy;

в) симметричную Х относительно начала координат;

г) получающуюся из Х поворотом вокруг начала координат на 90° против часовой стрелки.

24. Придумайте способ кодировки алгоритмов управления Чертежником с помощью одной литерной величины **лит** t.

25. Придумайте способ кодировки позиции в игре "крестики-нолики" на доске 10×10 клеток. Как подсчитать количество крестиков на доске? Как узнать, выиграли ли крестики?

26. Придумайте, как закодировать состояние в упражнении 1 § 3 (волк, коза и капуста). Как проверить, допустимо ли это состояние (т. е. что никто никого не съест)?

27. Составьте информационную модель расписания уроков а) одного ученика; б) одного класса (на некоторых уроках класс может делиться на группы); в) всей школы. Как подсчитать число уроков в среду (модели а и б)? Как узнать, сколько раз за неделю класс переходит из одной комнаты (кабинета) в другую (модель б)? Сколько учителей в школе (модель в)? Как определить, сколько уроков в неделю проводит данный учитель (модель в)?

28. На складе хранятся приборы 1000 наименований, не более 100 экземпляров каждого прибора. Приборы размещены в 50 помещениях, вмещающих до 2000 приборов каждое. Составьте информационную модель склада, позволяющую узнать, есть ли на складе данный прибор и в каком помещении его искать.

29. Составьте информационные модели:

а) свидетельства о рождении;

б) паспорта;

в) свидетельства о браке;

г) аттестата о среднем образовании;

д) авиационного билета.

Как проверить, могут ли все 5 перечисленных документов принадлежать одному и тому же человеку?

30. В камере хранения 100 ячеек. При выдаче багажа взимается плата 30 к. за каждые полные или неполные сутки хранения. Придумайте информационную модель камеры хранения, позволяющую выяснять, есть ли свободные ячейки, и вычислять плату при выдаче багажа.

§ 22. ИНФОРМАЦИОННОЕ МОДЕЛИРОВАНИЕ ИСПОЛНИТЕЛЕЙ НА ЭВМ (ИСПОЛНИТЕЛИ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ)

22.1. ИНФОРМАЦИОННАЯ МОДЕЛЬ ИСПОЛНИТЕЛЯ РОБОТ

В предыдущем параграфе (см. 21. 5) мы построили информационную модель поля Робота (М16) и показали, как с ее помощью имитировать команды Робота. При этом, однако, мы составляли лишь фрагменты (тела) алгоритмов, а заголовки не писали. Дело в том, что для полного оформления алгоритма нам пришлось бы в качестве его аргументов и результатов указать все величины модели (М16). А это очень громоздко и некрасиво.

Для удобства информационного моделирования ***исполнителей*** в алгоритмическом языке есть специальная конструкция **исп** — **кон**, которая позволяет промоделировать одновременно и обстановку на поле Робота, и алгоритмы выполнения команд Робота. С помощью этой конструкции ***информационную модель исполнителя*** Робот (на поле 32 на 16 клеток) можно записать так:

```
исп маленький Робот
| цел x |1≤x≤32, x — координата Робота (слева направо)
| цел y |1≤y≤16, y — координата Робота (сверху вниз)
|
| лог таб  ниже [1:32,1:16] |  ниже [i, j]=да ↔ ниже стена
| лог таб  выше [1:32,1:16] |  выше [i, j]=да ↔ выше стена
| лог таб левее [1:32,1:16] |  левее [i, j]=да ↔ левее стена
| лог таб правее [1:32,1:16] | правее [i, j]=да ↔ правее стена
| лог таб  закр [1:32,1:16] |  закр [i, j]=да ↔ закрашена
|
| вещ таб a [1:32,1:16] |уровень радиации
| вещ таб t [1:32,1:16] |температура
|
| алг вправо                                              (A93)
| нач
|   x:=x+1
| кон
|
| алг лог справа стена                                    (A94)
| нач
|   знач :=правее [x, y]=да
| кон
| ... (алгоритмы, имитирующие остальные команды Робота)
кон
```

Информационная модель исполнителя Робот включает в себя:

а) уже известную нам информационную модель (М16) обстановки на поле Робота — она записывается после слова **исп**;

б) имитации команд Робота в этой модели — каждая команда исполнителя оформляется в виде отдельного алгоритма.

Величины модели x, y и другие не являются здесь ни аргументами, ни результатами, ни промежуточными величинами алгоритмов. Это еще один ***вид величин*** алгоритмического языка — ***общие величины исполнителей***. Как происходит работа с ними и как ЭВМ располагает их в памяти, мы рассмотрим в п. 22.5.

Благодаря использованию общих величин алгоритмы "вправо", "справа стена" и др. не имеют ни аргументов, ни результатов. Их вызовы внешне неотличимы от вызовов команд Робота. Поэтому алгоритмы работы с такой моделью ***ничем не отличаются*** от алгоритмов работы с настоящим исполнителем.

22.2. ИСПОЛНИТЕЛИ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

В общем случае информационная модель исполнителя в алгоритмическом языке записывается так:

```
исп имя
| описание общих величин исполнителя
| последовательность команд для задания начальных значений
|                                                общих величин
| последовательность алгоритмов исполнителя
кон
```

После служебного слова **исп** (исполнитель) записывается имя исполнителя. Общие величины описываются обычным образом, например:

цел x, y, **вещ таб** a [1:32,1:16], t [1:32,1:16]

Общие величины исполнителя могут использоваться внутри любых алгоритмов исполнителя, но не могут использоваться вне исполнителя.

Поскольку вызовы вспомогательных алгоритмов в алгоритмическом языке внешне неотличимы от вызовов команд исполнителей, то можно говорить не про алгоритмы исполнителя, а про его "команды", про вызовы этих команд и т. п. При составлении алгоритмов вообще не важно, существует ли исполнитель на самом деле или имитируется на ЭВМ. Поэтому мы будем говорить "исполнитель", не уточняя, идет ли речь о настоящем исполнителе или о его информационной модели.

22.3. ИСПОЛЬЗОВАНИЕ ИСПОЛНИТЕЛЕЙ ПРИ СОСТАВЛЕНИИ АЛГОРИТМОВ

Рассмотрим исполнителя, назовем его И1, который умеет выполнять три команды:

```
          запомнить (арг вещ x)
цел       число больших (арг вещ a0)
вещ       минимум
```

При выполнении команды "запомнить" исполнитель И1 запоминает аргумент команды в своей памяти. По команде "число больших (a0)" исполнитель И1 просматривает все запомненные числа и определяет, сколько среди них чисел больших, чем a0. По команде "минимум" И1 сообщает минимальное из запомненных чисел.

Исполнитель И1 можно использовать при решении самых разных задач. Например, можно подсчитать число клеток коридора, уровень радиации в которых выше, чем в клетке выхода:

```
алг число опасных клеток (рез цел n)                    (A95)
  дано |Робот в левой клетке горизонтального коридора длины≤100
  надо |Робот вышел из коридора вправо, n= число клеток,
       |уровень радиации в которых выше, чем в клетке выхода
нач
| нц пока снизу стена
| | запомнить (радиация); вправо
| кц
| n:= число больших (радиация)
кон
```

22.4. ЗАДАНИЕ ИСПОЛНИТЕЛЯ И1 НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

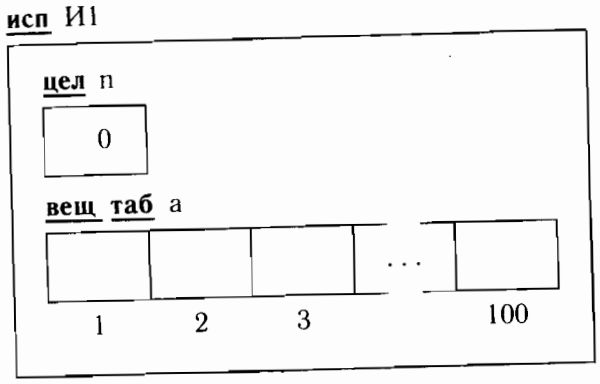
Исполнитель И1 пока существует только в нашем воображении — ведь мы его придумали. Если мы действительно хотим поручить ЭВМ выполнение алгоритма А95, то надо либо изготовить реального исполнителя И1 и подключить его к ЭВМ; либо — что гораздо проще — подменить исполнителя И1 его информационной моделью (т. е. заставить ЭВМ выполнять еще и команды исполнителя И1). Второй подход вместо изготовления И1 "в металле" требует лишь информационного ***описания*** И1 на алгоритмическом языке:

```
исп И1
| вещ таб a[1:100]
| цел n |количество запомненных чисел
| n:=0
|
| алг запомнить (арг вещ x)                              (A96)
|  дано n<100
|  надо |к запомненным числам добавлено число x
| нач
| | n:=n+1; a[n]:=x
| кон
|
| алг цел число больших (арг вещ a0)                     (A97)
|  надо |знач=количество чисел больших, чем a0
| нач цел i
| | знач :=0
| | нц для i от 1 до n
| | | если a[i]>a0 то знач:=знач+1 все
| | кц
| кон
|
| алг вещ минимум                                        (A98)
|  дано n>0
|  надо |знач=минимальное из запомненных чисел
| нач цел i
| | знач:=a[1]
| | нц для i от 2 до n
| | | если a[i]<знач то знач :=a[i] все
| | кц
| кон
кон
```

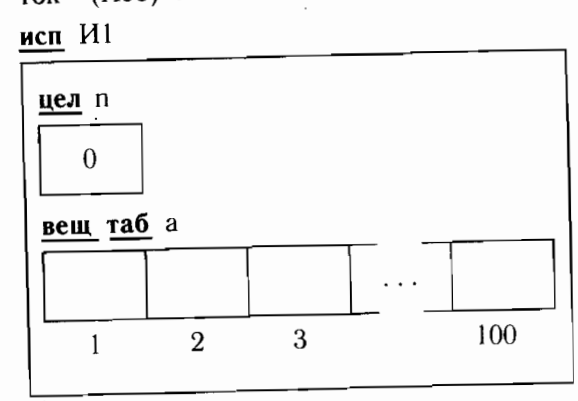

22.5. КАК ЭВМ РАБОТАЕТ С ОБЩИМИ ВЕЛИЧИНАМИ В ИНФОРМАЦИОННОЙ МОДЕЛИ ИСПОЛНИТЕЛЯ

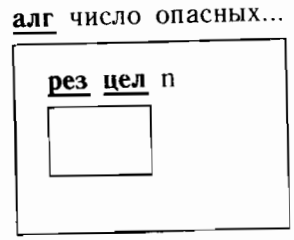
Будем, как и раньше, изображать память ЭВМ в виде классной доски. Опишем работу ЭВМ при выполнении алгоритма А95 с использованием информационной модели исполнителя И1 (п. 22.4).

До начала выполнения ЭВМ отведет место в памяти для хранения всех общих величин (в данном случае целочисленной величины n и вещественной таблицы a[1:100] исполнителя И1), а также выполнит команды, записанные между **исп** и **кон** *перед* алгоритмами (в данном случае одну команду "n:=0"):



Далее ЭВМ начнет выполнять алгоритм "число опасных клеток" (А95) и вначале отведет память для этого алгоритма.

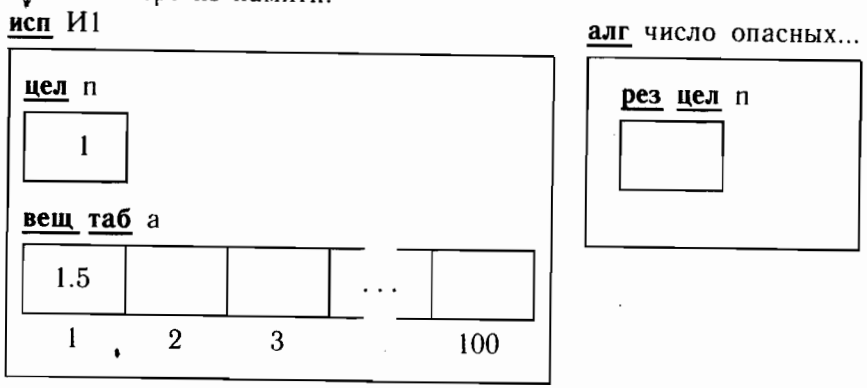




запомнить (1.5)

Предположим, что уровень радиации в первой клетке коридора равен 1.5. Тогда первым обращением к исполнителю И1 будет вызов "запомнить (1.5)". При выполнении этого вызова ЭВМ отведет память для алгоритма "запомнить" (А96) и начнет выполнять его:

В процессе выполнения алгоритма "запомнить" ЭВМ увеличит общую величину n на 1 и запомнит в a [1] значение аргумента x, т. е. 1.5. По окончании выполнения алгоритма "запомнить" его прямоугольник будет стерт из памяти:



При следующем вызове команды "запомнить" величина n в памяти исполнителя И1 будет увеличена еще на 1 и в элементе a [2] будет запомнен уровень радиации во второй клетке и т. д.

По окончании выполнения алгоритма "число опасных клеток" его прямоугольник будет стерт, а прямоугольник исполнителя И1 останется — общие величины ЭВМ хранит в памяти постоянно.

Таким образом, каждому исполнителю в алгоритмическом языке соответствует "большой прямоугольник" в памяти ЭВМ, внутри которого ЭВМ располагает общие величины исполнителя. Однако в отличие от прямоугольников алгоритмов прямоугольники исполнителей появляются в памяти ЭВМ ***до начала выполнения*** алгоритмов и существуют в течение всего времени выполнения. Меняя и используя значения общих величин, алгоритмы исполнителя могут обмениваться информацией друг с другом (этим они и отличаются от обычных вспомогательных алгоритмов).

22.6*. ИСПОЛЬЗОВАНИЕ ИСПОЛНИТЕЛЕЙ ПРИ РЕШЕНИИ ЧИСТО ИНФОРМАЦИОННЫХ ЗАДАЧ

Информационные модели исполнителей могут использоваться и при решении чисто информационных задач, когда в условии задачи никакие исполнители вообще не фигурируют. Пусть, например, в прямоугольной таблице чисел **вещ таб** t [1:60,1:60] требуется в каждом столбце найти максимальное число и среди этих 60 максимальных чисел найти минимальное. Используя И1, это можно сделать так:

```
алг вещ минимакс (арг вещ таб t [1:60,1:60])                (А99)
 надо |в каждом столбце найти максимальное число и установить
      |знач равным минимальному из них
нач цел j
| нц для j от 1 до 60
| | запомнить (макс (t, j))
| кц
| знач:=минимум
кон
```

Здесь "запомнить" и "минимум" — вызовы команд исполнителя И1, а "макс (t, j)" — вызов вспомогательного алгоритма-функции, который должен вычислять максимальное значение в j-м столбце таблицы t. Этот алгоритм-функцию можно записать так:

```
алг вещ макс (арг вещ таб t [1:60,1:60], цел j)              (А100)
 надо |знач=максимальное значение в j-м столбце таблицы t
нач цел i
| знач :=t [1, j]
| нц для i от 2 до 60
| | знач :=max (знач, t [i, j])
| кц
кон
```

22.7*. МЕТОД ПОСЛЕДОВАТЕЛЬНОГО УТОЧНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ИСПОЛНИТЕЛЕЙ

Метод последовательного уточнения применим и в случае использования вспомогательных исполнителей. Иными словами:

при составлении алгоритма можно использовать вызовы команд еще не существующих исполнителей, а потом описать информационные модели этих исполнителей на алгоритмическом языке.

Разумеется, при составлении алгоритмов такого ***вспомогательного*** исполнителя можно придумать новых (более простых) исполнителей и т. д. В самом конце этой цепочки могут быть (а могут и не быть) реальные исполнители (Робот, ядерный реактор, самолет и пр.). При выполнении ЭВМ просто будет вызывать из одних алгоритмов другие, имитируя всех придуманных нами исполнителей в соответствии с их описаниями на алгоритмическом языке.

Такой метод составления алгоритмов — ***метод последовательного уточнения с использованием исполнителей*** — в настоящее время является ***основным методом алгоритмизации сложных задач***.

УПРАЖНЕНИЯ

1. Составьте алгоритмы для исполнителя "маленький Робот", имитирующие все команды Робота.

2. Измените исполнителя "маленький Робот", используя для кодировки стен на поле Робота ваше решение упражнения 17 предыдущего параграфа.

3. Добавьте в исполнитель "И1" команды

а) "забыть все", при выполнении которой исполнитель должен стирать все запомненные числа из своей памяти;

б) "**лог** среди запомненных есть (**арг** **вещ** а0)", которая должна отвечать **да**, если среди запомненных чисел есть число а0, и **нет** в противном случае;

в) "**цел** число равных (**арг** **вещ** а0)", которая должна выдавать количество чисел среди запомненных, равных а0;

г) "стереть (**арг** **вещ** x)", при выполнении которой из памяти стирается любое из запомненных чисел, равных x (если такого числа в памяти И1 нет, то память должна остаться прежней, но ни отказа, ни зацикливания возникать не должно).

4. Измените исполнителя И1 так, чтобы получился исполнитель И2 со следующей системой команд:

	запомнить (**арг** **вещ** x)	число добавляется к запомненным
цел	число запомненных	
	забыть все	
вещ	последнее	последнее запомненное число
	сложить два последних	
	умножить два последних	

При выполнении команд сложения и умножения И2 должен два последних числа заменить их суммой или произведением (количество запомненных чисел при этом уменьшится на 1).

5. Известно, что исполнитель И2 (упр. 4) запомнил коэффициенты квадратного трехчлена, начиная со свободного члена. Составьте алгоритм, вычисляющий значение этого трехчлена в точке x (состояние И2 после выполнения алгоритма может быть любым).

6. Известно, что исполнитель И2 (упр. 4) запомнил коэффициенты некоторого многочлена, начиная со свободного члена. Составьте алгоритм, вычисляющий значение этого многочлена в точке x (состояние И2 после выполнения алгоритма может быть любым).

У к а з а н и е. Используйте схему Горнера вычисления значения многочлена в точке.

7. Известно, что исполнитель И2 (упр. 4) запомнил 10 чисел. Составьте алгоритм, который найдет первое запомненное число (состояние И2 после выполнения алгоритма может быть любым).

8. Используя исполнителя Чертежник, запишите на алгоритмическом языке ***модель исполнителя Черепашка*** со следующими командами:

поднять хвост	вперед (**арг** **вещ** r)	направо (**арг** **вещ** α)
опустить хвост	назад (**арг** **вещ** r)	налево (**арг** **вещ** α)

В начальный момент Черепашка "смотрит" прямо вверх (вдоль оси y) и хвост у нее поднят. По команде "вперед (r)" Черепашка ползет вперед расстояние r. Если ее хвост поднят, то никаких следов не остается. Если хвост опущен, то за Черепашкой остается след — отрезок длины r. По команде "назад r" Черепашка пятится назад на расстояние r. По команде "направо (α)" Черепашка поворачивается вокруг хвоста на α градусов по часовой стрелке, по команде "налево (α)" — против часовой стрелки. Хвост Черепашки при поворотах остается на месте.

9. Опишите, как будут меняться общие величины исполнителя Черепашка (упр. 8) в процессе выполнения алгоритма

(А101)

```
алг след
 надо |на песке нарисован квадрат со стороной 1
нач
| опустить хвост
| вперед (1); направо (90)
| вперед (1); направо (90)
| вперед (1); направо (90)
| вперед (1); направо (90)
| поднять хвост
кон
```

10. Нарисуйте след, который оставит Черепашка (упр. 8) после выполнения команд:

а)
```
нц для k от 1 до 10
| вперед (k)
| направо (90)
кц
```

б)
```
нц для k от 1 до 10
| вперед (1)
| направо (90)
| вперед (k)
| направо (90)
кц
```

в)
```
нц для k от 1 до 10
| вперед (k)
| направо (90)
| вперед (k+1)
| направо (90)
кц
```

11. Составьте алгоритм для Черепашки (упр. 8), после выполнения которого будут нарисованы:

а) спираль из n звеньев;
б) правильный шестиугольник;
в) правильный пятиугольник;
г) пятиугольная звезда;
д) правильный n-угольник со стороной 1.

§ 23. ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Информационные системы — это системы, предназначенные для хранения большого объема информации; быстрого поиска требуемой информации и вывода ее в удобном для человека виде; добавления, удаления и изменения хранимой информации.

23.1. СИСТЕМА ПРОДАЖИ ЖЕЛЕЗНОДОРОЖНЫХ БИЛЕТОВ ЭКСПРЕСС

Система имеет несколько центров. Московский центр обслуживает около 800 железнодорожных касс и справочных бюро внутри Москвы, а также еще около 40 городов.

Система содержит информацию о сети железных дорог СССР и информацию обо всех поездах дальнего следования, в том числе:

— расписание движения поездов — их номера, откуда, куда и когда идет каждый поезд, где он останавливается и т. п.;

— состав каждого поезда — количество вагонов, их типы (купейный, плацкартный, общий и пр.);

— информацию о том, на какие места билеты уже проданы, а какие места свободны.

Суммарный объем этой информации составляет около 40 Мбайт. Эта информация с течением времени меняется, например при продаже и возврате билетов. Архив системы, содержащий информацию обо всех изменениях за сутки, имеет объем около 70 Мбайт.

Кроме информации о поездах, система ЭКСПРЕСС хранит информацию о количестве денег, полученных за каждую смену в каждой кассе, о количестве израсходованных проездных бланков (билетов), а также некоторую другую информацию. Объем этой информации (за сутки) составляет около 10 Мбайт.

Всего при работе системы для хранения информации используются два магнитных диска по 100 Мбайт каждый.

Пользователями системы ЭКСПРЕСС являются кассиры (продажа и возврат билетов), работники справочных бюро и пассажиры (справки о наличии мест), финансовые работники (получение финансовых отчетов о работе касс), административный персонал (изменение расписания, сбор статистической информации). Система продает за день в среднем 130 тыс. мест на 1.5 млн. рублей, обслуживая в часы пик по 12—15 заявок в секунду.

23.2. ИНФОРМАЦИОННО-УПРАВЛЯЮЩАЯ СИСТЕМА ВОЛЖСКОГО АВТОЗАВОДА

Волжский автомобильный завод в г. Тольятти ежегодно выпускает около миллиона легковых автомобилей "Жигули", "Лада", "Нива", "Спутник". Сборочный конвейер этого завода работает под управлением ЭВМ, которая обеспечивает своевременное поступление деталей на конвейер со складов и из цехов вспомогательных производств. Для выполнения этой задачи информационно-управляющая система хранит информацию о тысячах узлов и деталей, из которых собирается автомобиль, о запасах деталей на складах, об их движении по транспортным линиям и т. д. На основе этой информации ЭВМ самостоятельно управляет автоматизированными складами, транспортными конвейерами, а также рядом других устройств.

23.3. ИНФОРМАЦИОННО-УЧЕТНАЯ СИСТЕМА МЕЖДУГОРОДНОЙ ТЕЛЕФОННОЙ СВЯЗИ

В настоящее время во многих городах нашей страны действует автоматическая междугородная телефонная связь: чтобы позвонить в другой город, достаточно со своего телефона набрать код города и номер телефона в этом городе. Для каждого разговора информационно-учетная система фиксирует его продолжительность, запоминает номера телефонов говорящих, подсчитывает стоимость разговора в соответствии с тарифом (в зависимости от расстояния между городами), готовит и печатает счет на оплату разговора, который потом посылается абоненту.

23.4. БАЗЫ ДАННЫХ

Для чисто информационной работы — хранения, изменения и обработки больших объемов взаимосвязанной информации — ис-

пользуются специальные системы, называемые ***базами данных***. Между базами данных и информационными системами нет строгой границы — в каждой информационной системе используется та или иная специализированная база данных.

В базе данных предприятия, например, может храниться вся информация о штатном расписании, рабочих и служащих предприятия, сведения о материальных ценностях, поступлении сырья и комплектующих, запасах на складах, выпуске готовой продукции, приказах и распоряжениях дирекции и т. д. Даже небольшие изменения какой-то одной информации могут приводить к значительным изменениям в разных других местах. Например, издание приказа о повышении одного работника в должности приводит к изменениям не только в личном деле данного работника, но и к изменениям в списках подразделения, в котором он работает, в ведомостях на зарплату, в данных о зарплате всего предприятия, в графике отпусков и т. п.

Существуют также ***распределенные базы данных***, в которых информация хранится в разных местах на разных ЭВМ, а для получения той или иной информации ЭВМ связываются между собой с помощью линий связи. (Все в целом это называется ***информационной сетью***.)

23.5. УЧЕБНАЯ ИНФОРМАЦИОННАЯ СИСТЕМА «ВАГОН»

Методы хранения и обработки информации в реальных информационных системах достаточно сложны. Трудозатраты на создание таких систем измеряются десятками и даже сотнями человеко-лет. Вместе с тем общие идеи этих систем можно понять на простых, "игрушечных" примерах.

Рассмотрим информационную систему "Вагон" для хранения информации о свободных и занятых местах в одном 36-местном купированном вагоне, следующем без остановок по маршруту Москва — Петушки. Как создать такую систему? Прежде всего построим информационную модель вагона:

```
исп Вагон
| цел таб M[1:36] |M[i]=0 ↔ место i свободно
                  |M[i]=1 ↔ место i продано
```

Теперь покажем, как могут выглядеть алгоритмы, использующие общую величину M и выполняющие обработку информации в системе "Вагон":

```
| алг учесть продажу места (арг цел i)                    (A102)
|  дано 1⩽i⩽36 и M[i]=0
| нач
|  M[i]:=1
| кон
```



```
| алг учесть возврат билета на место (арг цел i)          (A103)
|  дано 1⩽i⩽36 и M[i]=1
| нач
|  M[i]:=0
| кон

| алг цел количество проданных мест                       (A104)
| нач цел i
|  знач:=0
|  нц для i от 1 до 36
|   знач:=знач+M[i]
|  кц
| кон

| алг цел номер свободного места                          (A105)
|  дано |свободные места есть
| нач
|  знач:=1
|  нц пока M[знач]≠0
|   знач:=знач+1
|  кц
| кон
| ... ... ...
кон
```

Мы рассмотрели работу с одним вагоном. Если требуется закодировать поезд, состоящий из 14 таких вагонов, то можно, например, использовать прямоугольную таблицу

```
цел таб M[1:14,1:36]
```

где первый индекс означает номер вагона, а второй — номер места в вагоне.

23.6. УЧЕБНАЯ ИНФОРМАЦИОННАЯ СИСТЕМА «ТЕЛЕФОННАЯ КНИЖКА»

Рассмотрим еще один пример "игрушечной" информационной системы. Пусть требуется создать справочную систему обо всех владельцах телефонов в маленьком городе (число телефонов меньше 1000). Эту информацию можно закодировать так:

```
исп Телефонная книжка
| цел n                  |число телефонов в городе
| лит таб ФИО[1:999]     |ФИО владельца телефона с номером i.
                         |Если телефона i нет, то ФИО[i]=""
```

Приведем несколько алгоритмов из этого исполнителя:

```
алг учесть установку нового телефона (арг цел t, лит f)        (A106)
 дано ФИО[t] = "" |t — номер телефона, f — ФИО абонента
 надо |информация о новом телефоне запомнена
нач
| n:=n+1
| ФИО[t]:=f
кон

алг смена номера (арг цел t стар, t нов)                       (A107)
 дано ФИО [t нов] = "" |t стар — старый номер, t нов — новый
 надо |соответствующая информация изменена
 нач
| ФИО[t нов]:=ФИО[t стар]
| ФИО [t стар]:= ""
кон
```

А вот как может выглядеть алгоритм, который запрашивает у человека ФИО абонента и выдает в ответ номер его телефона:

```
алг справка о номере телефона                                  (A108)
нач лит f, цел t
| вывод "Укажите ФИО абонента: "; ввод f
| поиск абонента (f, t)
| если t=0
| | то    вывод "Абонент не найден"
| | иначе вывод "Абонент", f, "имеет телефон", t
| все
кон
```

Здесь использован вспомогательный алгоритм "поиск абонента", который по значению элемента f должен определить его индекс t в таблице ФИО. Этот алгоритм аналогичен алгоритму A67:

```
алг поиск абонента (арг лит f, рез цел t)                      (A109)
 надо |ФИО[t]=f, если значение f встречается в таблице ФИО,
      |t=0 в противном случае
нач цел i
| t:=0
| нц для i от 1 до 999
| | если ФИО[i]=f то t:=i все
| кц
кон
...
кон
```


УПРАЖНЕНИЯ

1. Добавьте в исполнитель "Вагон" следующие команды:
а) начать работу (все места должны стать свободными);
б) лог есть место на нижней полке;
в) найти два свободных места в одном купе (рез цел n1, n2).

2. Пусть информация о проданных билетах и свободных местах в поезде хранится в таблице

лог таб свободно [1:14, 1:36]

(свободно [i, j] = да, если j-е место в i-м вагоне свободно). Выполните упражнение 1 применительно к поезду (в алгоритме 1в добавьте еще один результат — номер вагона).

3. Придумайте информационную модель поезда, в котором первые 8 вагонов — купейные, 9-й — вагон-ресторан, а вагоны с 10 по 14 — плацкартные. Пусть ваш класс в полном составе собрался поехать на экскурсию. Сформулируйте требования, которым, по вашему мнению, должны удовлетворять места в поезде для такой поездки (например, "все места в одном вагоне", "не более чем в двух соседних вагонах", "рядом с рестораном" и т. п.). Составьте алгоритм, который определит, есть ли в поезде нужное количество мест, удовлетворяющих вашим требованиям.

4. Пусть в системе "Телефонная книжка" информация о владельцах телефонов задается следующими общими величинами:

```
цел n                    |общее количество телефонов
лит таб ФИО [1:999]      |ФИО владельца i-го телефона
цел таб     t [1:999]    |номер        i-го телефона
```

а) измените алгоритмы A106—A109 для этой информационной модели;

б) пусть абоненты в таблице ФИО уже упорядочены по алфавиту. Составьте алгоритм установки нового телефона так, чтобы после его выполнения свойство упорядоченности таблицы ФИО сохранялось.

5. В рамках исполнителя "Телефонная книжка" придумайте способ кодирования информации об адресах абонентов. Составьте алгоритм, который по адресу дома определяет количество телефонов, установленных в этом доме.

6. Придумайте способ кодирования информации и опишите общие величины исполнителя "Автомобили", предназначенного для хранения информации об автомобилях (номер, ФИО владельца, марка автомашины и ее цвет). Составьте алгоритм, выводящий на экран список всех владельцев белых "Волг", в номерах которых есть буква М и цифра 7.

§ 24. ОБРАБОТКА ТЕКСТОВОЙ ИНФОРМАЦИИ

При взаимодействии человека с ЭВМ информация может быть представлена в разных формах: в виде изображения или текста на экране или на бумаге, в виде мелодии или речи и т. п. В этом параграфе мы познакомимся с системами обработки текстовой информации.

24.1. СИСТЕМЫ ОБРАБОТКИ ТЕКСТОВ

Для ввода, изучения и изменения текстовой информации применяются специальные ***системы обработки текстов*** (часто их называют ***текстовыми редакторами***).

Основное достоинство обработки текстов на ЭВМ — легкость изменения, размножения и копирования информации. При внесении изменений в большой текст основная часть его остается нетронутой — изменения затрагивают только то, что действительно надо менять. Если некоторый фрагмент текста уже введен, то его можно скопировать в другие тексты, не вводя заново. В любой момент, хотя бы и через год после ввода текста, можно получить его новую копию, напечатать в необходимом количестве экземпляров и т. п.

ЭВМ позволяет автоматизировать операции над текстами, например заменить во всем тексте одно слово на другое (сколько бы раз оно ни встречалось), быстро найти интересующий фрагмент текста или все ссылки на какой-то рисунок.

Посмотрите, например, на текст, который вы читаете,— он аккуратно выровнен слева и справа. Представьте, что в этот красиво расположенный текст надо вставить пару слов — весь текст "поплывет": слова, которые раньше переносились со строки на строку, могут оказаться в середине, а слова из середины, наоборот, могут попасть на границы строк. Для получения после вставки "ровного" текста надо рассчитать новое положение слов и расставить их по строкам. Эту нетворческую, техническую работу (она называется ***форматированием текста***) без труда выполняют современные системы обработки текстов.

Еще одна приятная возможность при работе с текстом на ЭВМ — ***откатка*** изменений ("вспять по времени"), т. е. восстановление предыдущих состояний текста. С помощью откатки можно восстановить ошибочно удаленные строки, абзацы или разделы, вернуться к предыдущему варианту текста и т. п.

24.2. ТЕКСТ, КУРСОР И ОКНО

Обычно при работе с системами обработки текстов экран служит как бы ***окном***, через которое человек глядит на текст. Текст при этом удобно представлять в виде длинного бумажного свитка, расположенного "за окном" (рис. 83, *а*).

Специальный значок ***(курсор)*** показывает место в тексте, в котором можно вставлять, удалять или изменять буквы.

Курсор можно двигать по тексту, нажимая на клавиши со стрелками. При перемещении за пределы окна курсор "упирается" в край окна и сдвигает его (рис. 83). Конечно, экран монитора при этом остается на месте — просто меняется изображаемый текст. При движении курсора вниз текст смещается по экрану вверх, а при движении курсора вверх текст смещается вниз.

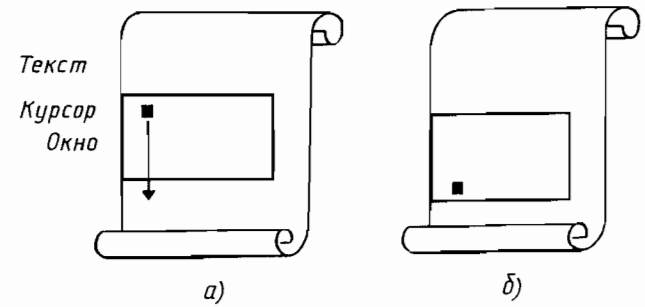


Рис. 83

Иногда человеку бывает нужно работать одновременно с несколькими текстами. В этом случае на экране изображается несколько окон (фото 18 вклейки) и в каждом окне изображается свой текст.

24.3. УЧЕБНАЯ МОДЕЛЬ РЕДАКТОРА ТЕКСТОВ

Рассмотрим простейший случай. Пусть текст целиком помещается в оперативной памяти ЭВМ и состоит не более чем из 200 строк размером не более 100 символов каждая, а окно вмещает 24 строки по 80 символов. Построим информационную модель текста, окна и курсора (рис. 84):

```
исп Простейший редактор
| цел N                          | число строк в тексте
| сим таб T [1:200, 1:100]       | i-я строка хранится в T [i, 1:100]
|
| цел Y0, X0  | положение окна:
|             | Y0 — число строк текста, расположенных выше окна
|             | X0 — число позиций (символов) текста левее окна
|
| цел Yk, Xk  | координаты курсора в окне:
|             | 1 ⩽ Yk ⩽ 24 — номер строки (сверху вниз),
|             | 1 ⩽ Xk ⩽ 80 — номер позиции в строке
```

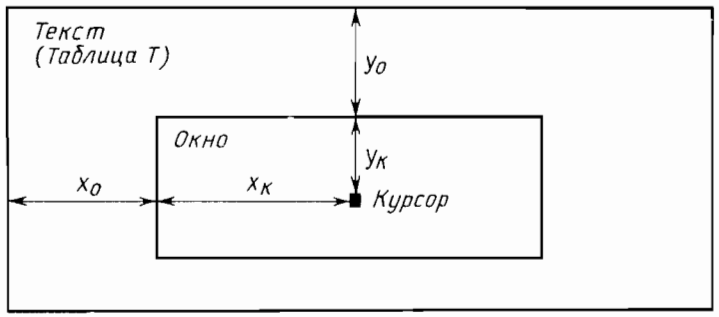


Рис. 84

Зная значения величин Y0 и Yk, легко определить номер строки текста, на которой стоит курсор,— это y=Y0+Yk. Аналогично, позиция курсора в тексте — это x=X0+Xk. Также легко узнать, какие строки видны на экране: первая строка имеет номер Y0+1, последняя — Y0+24 (конечно, если такая строка есть, т. е. если Y0+24≤N). В каждой строке изображается 80 символов: от X0+1 до X0+80. Приведем пару алгоритмов:

```
алг вниз на строку                                              (A110)
 надо | курсор смещен вниз на строку, если нужно, сдвинуто
      | и окно. Если курсор уже был за последней строкой
      | текста и сдвинуть его нельзя, то звонок
нач
 выбор
  при Y0+Yk>N: звонок       | курсор ниже текста ⇒ звонок
  при Yk<24  : Yk:=Yk+1     | сдвиг курсора внутри окна
    иначе      Y0:=Y0+1     | курсор "уперся"⇒сдвиг окна
 все
кон
```

Стандартная функция алгоритмического языка "звонок", которую мы здесь использовали, приводит к выдаче звукового сигнала.

```
алг удалить символ                                              (A111)
 надо | удален символ текста, на котором стоял курсор. Сим-
      | волы правее курсора сдвинуты на одну позицию влево,
      | справа добавлен пробел
нач цел y, i
 y:=Y0+Yk                  | номер строки в таблице T
 нц для i от X0+Xk до 99   | сдвиг символов правее курсора
  T [y, i]:=T [y, i+1]     |   на одну позицию влево
 кц                        |
 T [y, 100]:=" "           | справа добавляется один пробел
кон
```


В алгоритмах A110 и A111 меняются только значения общих величин информационной модели. При работе с редактором текстов эти изменения надо еще отобразить на экране:

```
алг переизобразить экран и курсор                               (A112)
нач цел m, i
 очистить экран
 m:=min (24, N−Y0)  | число строк текста, попавших в окно
 нц для i от 1 до m
  поз (i, 1); изобразить строку (Y0+i)
 кц
 поз (Yk, Xk)       | изображение курсора в нужной позиции
кон
```

Здесь используются вспомогательный алгоритм "очистить экран" (см. § 20, упр. 2), команда исполнителя Экран "поз" (см. п. 20.2) и вспомогательный алгоритм "изобразить строку":

```
алг изобразить строку (арг цел y)                               (A113)
 дано | y — номер строки в таблице T
 надо | на экране изображена попадающая в окно часть строки,
      | т. е. символы от X0+1 до X0+80
нач цел j
 нц для j от X0+1 до X0+80
  вывсим (T [y, j])   | команда вывсим описана в п. 20.2
 кц
кон

 . . .  (алгоритмы других команд редактора)
кон
```

Итак, мы разобрали информационную модель текста и работу алгоритмов, которые его меняют, изображая изменения на экране. Кроме этой основной части, записанной на алгоритмическом языке с помощью конструкции "исполнитель", текстовый редактор включает в себя основной алгоритм (его мы не приводим), при выполнении которого ЭВМ ждет, пока человек нажмет на клавишу (команда "клав" п. 20.3); затем вызывает (выполняет) соответствующий этой клавише алгоритм (например, A110, если человек нажал на стрелку вниз); изображает результат на экране (т. е. вызывает алгоритм A112); снова ждет, пока человек нажмет клавишу, и т. д. Этот процесс прекращается по специальной команде окончания редактирования. Кроме того, основной алгоритм читает текст с диска перед началом и записывает измененный текст на диск в конце работы.

УПРАЖНЕНИЯ

1. В рамках исполнителя "Простейший редактор" составьте алгоритмы со следующими заголовками:

а) **алг** курсор в начало строки
надо | курсор в начале строки

б) **алг** удалить конец строки
надо | удалены все символы строки правее курсора
| (т. е. заменены на пробелы)

в) **алг** в начало строки по непробелам
надо | курсор на месте первого непробела в строке

г) **алг** курсор в конец строки по непробелам
надо | курсор за последним непробелом в строке

д) **алг** курсор влево на позицию
надо | курсор смещен на одну позицию влево (если он
| уже был в первой позиции, то звонок)

е) **алг** курсор вверх на строку
надо | курсор смещен на одну строку вверх (если он
| уже был в первой строке, то звонок)

ж) **алг** курсор в начало следующей строки
надо | курсор смещен вниз в начало следующей строки
| (если надо, сдвинуто и окно). Если курсор уже был
| за последней строкой текста, то звонок

з) **алг** центрирование строки
надо | текст в строке расположен примерно по центру, т. е.
| число пробелов слева и справа от текста отличается
| не более чем на 1

и) **алг** на слово вправо
надо | курсор смещен вправо в начало следующего слова,
| т. е. находится под первым символом слова

2. Придумайте другие команды текстового редактора. Составьте соответствующие этим командам алгоритмы (в рамках исполнителя "Простейший редактор").

3. Известно, что ширина текста не превышает 63 символов в строке. Как упростить исполнителя "Простейший редактор" для этого случая?

4. Придумайте какую-нибудь другую информационную модель текста, окна и курсора. Опишите преимущества и недостатки вашей модели по сравнению с моделью, приведенной в учебнике.

§ 25. НАУЧНО-ТЕХНИЧЕСКИЕ РАСЧЕТЫ НА ЭВМ

25.1. ЭВМ — ВЫЧИСЛИТЕЛЬНАЯ МАШИНА

Один из древнейших видов работы с информацией — вычисления. Земледельцам, строителям, мореплавателям, торговцам приходилось проводить те или иные подсчеты для определения, например, площади поля, размеров строительных блоков, местонахождения корабля и т. п. По мере развития науки и техники, по мере усложнения машин и сооружений потребность в вычислениях возрастала, они становились все более громоздкими. Достаточно сказать, что расчет запуска ракеты на нужную орбиту требует выполнения нескольких миллионов арифметических действий.

Именно потребности в автоматизации вычислений первоначально привели к появлению ЭВМ — электронных ***вычислительных*** машин. Часто используемое название ***компьютер*** также происходит от англ. compute — считать, вычислять. Вычисления, научно-технические расчеты были первой областью применений ЭВМ, и лишь позже ЭВМ начала применяться как универсальная машина для обработки информации.

Вычислительные возможности ЭВМ используются во многих применениях ЭВМ: при расчете зарплаты и прочностном расчете автомобиля, при управлении самолетом или ракетой, при расшифровке древних надписей и попытке узнать, о чем говорили гончарные мастера при изготовлении глиняных кувшинов. Использование уникальных вычислительных возможностей ЭВМ позволило создать ряд принципиально новых приборов и устройств, без ЭВМ вообще немыслимых. В этом параграфе мы расскажем только про одно из таких устройств — томограф.

25.2. ТОМОГРАФИЯ

Все вы знаете о "рентгене" — методе диагностики заболеваний с помощью просвечивания человека рентгеновскими лучами. На рентгеновском снимке врач видит "тень", отброшенную органами человека, и может безошибочно распознать трещины и переломы костей, вывихи, наличие посторонних предметов и т. д. Однако обнаружить, например, опухоль мозга с помощью рентгена практически невозможно, так как черепные кости экранируют внутренние ткани мозга. Поэтому вплоть до недавнего времени при подозрении на опухоль мозга врач вынужден был прибегать к вскрытию черепа пациента — иногда лишь для того, чтобы убедиться, что подозрения были напрасными!

Томограф (от греческого корня tomo — срез) позволяет врачу получить не "тень" от просвечивания, а изображение "среза" человеческого тела. Как устроен томограф? Обычно он представляет собой кольцо диаметром около метра, на котором укрепле-

но несколько сотен источников рентгеновских лучей и несколько сотен датчиков. Внутрь кольца помещается человек, после чего последовательно один за другим на десятые доли секунды включаются источники. При каждом включении очередного источника со всех датчиков снимаются показания — интенсивность рентгеновских лучей, прошедших через человеческий организм в разных направлениях (от источника к датчикам). После включения всех источников получаются сотни тысяч чисел. Эти числа сложным образом обрабатываются на ЭВМ, и ЭВМ изображает срез (плотность) человеческого тела (как если бы вместо измерений кольцо просто "разрезало" человека).

Обработка результатов измерений на ЭВМ требует выполнения нескольких миллиардов операций сложения и умножения над многозначными числами. Поэтому своим появлением на свет томография обязана прежде всего ЭВМ. Современные специализированные ЭВМ в состоянии изготовить томограмму за время порядка секунды.

Получив несколько соседних "срезов", можно с помощью ЭВМ синтезировать и трехмерное (объемное) изображение внутренних органов человека, опухолей и т. п. С помощью томографа можно не только диагностировать, но и лечить, например ввести лекарство точно в нужную точку и тем самым обойтись без хирургической операции. Наконец, сфера применения томографа не ограничивается только медициной. Вместо человека в томограф можно поместить техническое изделие, "диагностировать" в нем внутренние, скрытые дефекты, трещины и т. п.

25.3. ПРИБЛИЖЕННЫЕ ВЫЧИСЛЕНИЯ

Вычисления на ЭВМ обладают некоторой спецификой, отличающей их от привычных вам вычислений в математике. Рассмотрим, например, задачу вычисления корня уравнения $f(x)=0$. В курсе школьной математики вы рассматривали только такие уравнения, для которых ответ можно вычислить по формуле, например, для уравнения $ax^2+bx+c=0$:

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}.$$

Во многих случаях, однако, ответ не выражается формулой. Например, для корня уравнения $\cos(x)=x$ (т. е. для корня функции $f(x)=\cos(x)-x$) подобной простой формулы нет. В этом случае можно пытаться, ***не выводя точных формул***, вычислить корень ***приближенно***, например с точностью до 0.001. Несколько методов такого приближенного вычисления разных математических величин мы сейчас рассмотрим, и в частности один из методов вычисления корня. Подчеркнем, что мы будем вычислять

корень, не имея для него точных формул и не выводя их. Наш ответ будет неточным (приближенным), но зато мы сможем один и тот же метод применить к самым разным уравнениям, в том числе и к таким, которые математика точно решать не умеет.

25.4. ВЫЧИСЛЕНИЕ КОРНЯ ФУНКЦИИ МЕТОДОМ ДЕЛЕНИЯ ОТРЕЗКА ПОПОЛАМ

Пусть f — непрерывная функция и $f(a)\cdot f(b)\leqslant 0$. Тогда f обязательно имеет корень на отрезке $[a, b]$. Возьмем середину отрезка $c=(a+b)/2$ в качестве приближенного значения корня. Настоящий корень отличается от c не более чем на половину длины отрезка, т. е. не более чем на $(b-a)/2$. Если такая точность нас не устраивает, то можно от отрезка $[a, b]$ перейти к одной из его половин: либо к $[a, c]$, либо к $[c, b]$, а именно:

если $f(a)*f(c)\leqslant 0$, то корень на отрезке $[a, c]$;

если $f(c)*f(b)\leqslant 0$, то корень на отрезке $[c, b]$.

Так как длина нового отрезка вдвое меньше старого, то, взяв в качестве приближенного значения корня середину нового отрезка, мы получим корень с точностью $(b-a)/4$. Если и эта точность нас не устраивает, можно поделить пополам новый отрезок и т. д.

Таким образом, мы можем делить отрезок пополам и переходить к одной из его половин, пока длина отрезка не станет достаточно малой, а потом в качестве корня взять середину отрезка. Соответствующий алгоритм приближенного вычисления корня записывается так:

(A114)

```
алг вещ корень (арг вещ A, B, d)
 дано A<B и d>0 и F(A)*F(B)≤0
 надо | знач ≈ корень F(x)=0 на отрезке [A, B] с точностью d
нач вещ a, b, c
| a:=A; b:=B
| нц пока (b-a)/2>d | пока требуемая точность не достигнута
| | c:=(a+b)/2       | c:=середина отрезка [a, b]
| | если F(a)*F(c)≤0
| | | то b:=c        | переход к отрезку [a, c]
| | | иначе a:=c     | переход к отрезку [c, b]
| | все
| кц
|   утв F(a)*F(b)≤0 и (b-a)/2≤d
| знач:=(a+b)/2
кон
```

Чтобы выполнить этот алгоритм, надо задать вспомогательный алгоритм-функцию с именем F. Для уравнения $\cos(x)=x$, например, можно написать:

```
алг вещ F (арг вещ x)                                    (A115)
нач
| знач:=cos (x) — x
кон
```

После этого для вычисления корня уравнения cos (x) = x на отрезке [0, 2] с точностью 0.001 достаточно написать "x:=корень (0, 2, 0.001)".

25.5. ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ИНТЕГРАЛА МЕТОДОМ ТРАПЕЦИЙ

Для приближенного вычисления интеграла от функции на отрезке (т. е. для вычисления площади под графиком функции — рис. 85) разобьем отрезок [a, b] на n равных частей (например, на n = 10) и заменим график f (x) ломаной, состоящей из n звеньев (подобно тому, как мы поступали при рисовании параболы в § 11).

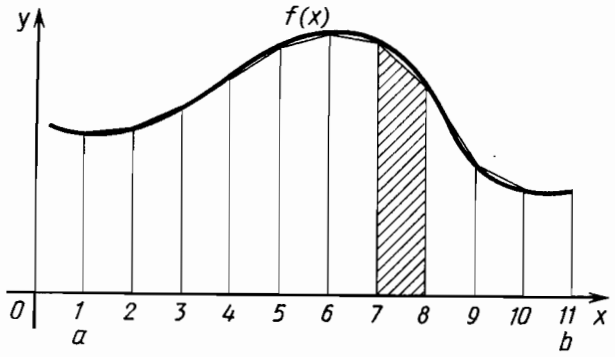


Рис. 85

Тогда искомую площадь можно приближенно вычислить как площадь под ломаной, т. е. как сумму площадей получившихся маленьких трапеций:

$$S \approx S_1 + S_2 + S_3 + \ldots + S_n,$$

где площадь одной трапеции, например S7, можно вычислить, умножив полусумму оснований f(x7) и f(x8) на высоту h = = (b — a)/n:

$$S_7 = \frac{f(x_7) + f(x_8)}{2} \cdot h.$$

Подставим выражения для S_1, S_2, ..., S_n в формулу для S:

$$S \approx \left(\frac{f(x_1)}{2} + f(x_2) + f(x_3) + \ldots + f(x_n) + \frac{f(x_{n+1})}{2} \right) \cdot h.$$

Обозначив сумму в скобках через s, алгоритм можно записать так:

```
алг вещ площадь (арг вещ a, b, цел n)                    (A116)
нач вещ s, h
| h:=(b—a)/n
| s:=f (a)/2+f (b)/2
| x:=a
| нц n—1 раз
| | x:=x+h   | переход к следующей точке
| | s:=s+f (x)
| кц
| знач:=s * h
кон
```

С увеличением n "зазор" между ломаной и графиком функции f (x) уменьшается и вычисленная с помощью алгоритма "площадь" величина становится все ближе к значению интеграла функции на отрезке.

25.6. МЕТОД МОНТЕ-КАРЛО

Рассмотрим еще один интересный метод приближенного вычисления площади — ***метод Монте-Карло***. Пусть у нас есть какая-нибудь фигура на плоскости, расположенная внутри стандартного квадрата со сторонами, параллельными координатным осям (рис. 86). И пусть про любую точку квадрата мы можем быстро узнать, попадает эта точка внутрь фигуры или нет. Тогда площадь можно вычислять так: засыплем квадрат ровным слоем песка. Ясно, что число песчинок, попавших внутрь фигуры, пропорционально ее площади: больше площадь — больше песчинок; меньше площадь — меньше песчинок. Поэтому, поделив количество песчинок, попавших внутрь фигуры, на количество всех песчинок в квадрате, мы можем найти, какую часть площади квадрата занимает фигура. Ну а площадь квадрата вычислить легко.

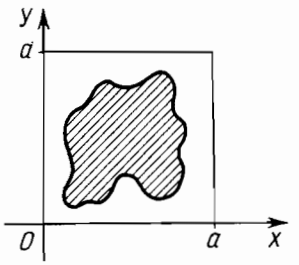


Рис. 86

Как записать эту идею на алгоритмическом языке? Что такое "песчинки" и как засыпать квадрат ровным слоем? Для этого используются так называемые ***случайные числа***. В алгоритмическом языке есть специальная стандартная функция rnd (a), значением которой является случайное число от 0 до a. Эта функция обычно называется ***датчиком случайных чисел***. При каждом вызове функции это число каждый раз свое. Если же вызвать функцию много раз подряд, то множество полученных чисел будет равномерно распределено по отрезку [0, a]. Пару случайных чисел можно рассматривать как координаты точки на плоскости (координаты "песчинки"). А алгоритм приближенного вычисления площади, использующий n "песчинок", запишется так:

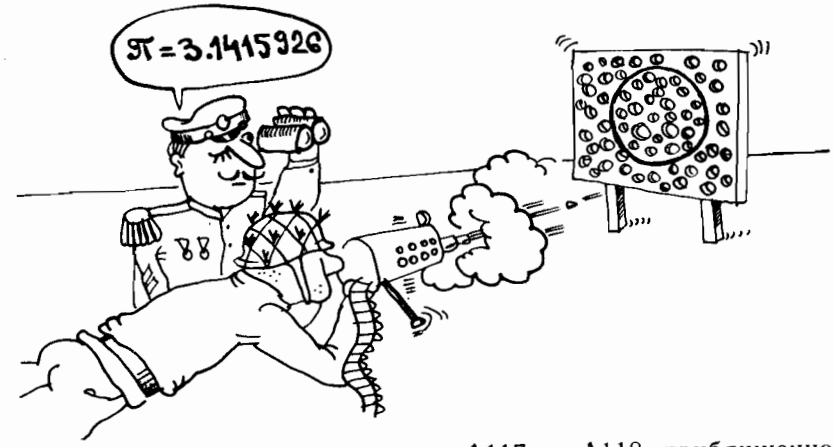
```
алг вещ площадь (арг цел n, вещ a)                    (A117)
 дано | n — количество "песчинок"
      | a — длина стороны квадрата (рис. 86)
 надо | знач ≈ площадь фигуры
нач цел m
| m:=0
| нц n раз
| | x:=rnd (a); y:=rnd (a)  | получение очередной "песчинки"
| | если точка внутри (x, y) | подсчет количества
| | то m:=m+1               | "песчинок", попавших
| | все                     |   внутрь фигуры
| кц
| знач:=(m/n) * a * a       | a * a — это площадь квадрата
кон
```

Здесь использован вспомогательный алгоритм "точка внутри", который по координатам точки (x, y) должен определять, попала точка внутрь фигуры или нет,— ведь мы предположили, что это мы делать умеем. Для каждой конкретной фигуры нужно составлять свой алгоритм "точка внутри". Взяв в качестве фигуры круг, можно применить алгоритм A117 для приближенного вычисления числа π.

25.7. ВЫЧИСЛЕНИЕ π МЕТОДОМ МОНТЕ-КАРЛО

Мы знаем, что площадь круга равна πR^2. Рассмотрим круг радиуса 1 с центром в точке (1, 1). Его площадь равна π. Но эту же площадь мы можем вычислить приближенно методом Монте-Карло, составив следующий алгоритм "точка внутри":

```
алг лог точка внутри (арг вещ x, y)                   (A118)
 надо | знач=точка (x, y) внутри круга радиуса 1
      | с центром в точке (1, 1)
нач
| знач:=((x-1) ** 2+(y-1) ** 2≤1)
кон
```



С использованием алгоритмов A117 и A118 приближенное вычисление числа π можно записать в виде команды присваивания: "пи:=площадь (n, 2)", где n — количество "песчинок". Поскольку используются случайные числа, то результаты могут быть разными даже при одном и том же n. Вот, например, некоторые результаты, полученные при выполнении этой команды авторами учебника:

n	10	100	500	1000	5000
пи	4.4	3.28	3.312	3.232	3.1424

УПРАЖНЕНИЯ

1. Как с помощью алгоритма "корень" (A114) найти:

а) корень уравнения $x \cdot \sin x - 1 = 0$ на отрезке $[0, 2\pi]$ с точностью до 0.02;

б) корень кубический из 3 с точностью 0.00001?

2. Как будет работать алгоритм "корень", если: а) функция имеет на заданном отрезке несколько корней; б) значения функции на концах отрезка одного знака; в) функция обращается в ноль точно посередине одного из уменьшающихся отрезков?

3. График функции

```
алг вещ F (арг вещ x)                                 (A119)
нач
| выбор
| | при x≥1 : знач:=x
| | при x<1 : знач:=x-2
| все
кон
```

имеет разрыв в точке x=1. Что произойдет при попытке выполнить для этой функции команду "x:=корень (0, 2, 0.001)"?

4. Какую величину вычисляет алгоритм (А116), если функция f может принимать и отрицательные значения?

5. Что получится при выполнении алгоритма (А116) для функции f (x) = |x| при a=−1, b=1, n=5?

6. Для каких функций алгоритм (А116) дает точный ответ?

7. Как вычислить π с использованием алгоритма (А116)?

8. По образцу алгоритма (А116) составьте алгоритм вычисления объема тела, получаемого при вращении вокруг оси *Ox* криволинейной трапеции, ограниченной прямыми x=a, x=b, графиком y=f (x) и осью *Ox*.

9. Автомобиль трогается с места и разгоняется в течение 4 с. Значения ускорения автомобиля в моменты времени t=0.1, t=0.2, t=0.3 и т. д. с шагом 0.1 с измеряются и записываются в таблицу **вещ таб** a [1:40]. Составьте алгоритм, приближенно вычисляющий расстояние, пройденное автомобилем до конца разгона, и скорость в конце разгона.

10. Четыре жука, расположенные в вершинах квадрата со стороной a, одновременно начинают ползти по направлению к следующему по часовой стрелке жуку с постоянной скоростью v. Составьте алгоритм, который с помощью Чертежника рисует приближенно траектории движения жуков за первые T секунд после начала движения.

У к а з а н и е. Замените траектории жуков ломаными — считайте, что в течение t секунд жуки ползут прямо (одно звено ломаной), потом поворачивают, опять t секунд ползут прямо и т. д.

11. По образцу (А118) составьте алгоритмы "точка внутри" для следующих фигур: а) полукруг; б) сектор; в) сегмент.

§ 26. МОДЕЛИРОВАНИЕ И ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ НА ЭВМ

26.1. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

ЭВМ — универсальная машина для обработки информации. Поэтому, если мы хотим про какую-то ситуацию что-то узнать (получить информацию!), мы можем попытаться ***смоделировать*** эту ситуацию на ЭВМ и получить информацию не из анализа реальной ситуации, а из анализа ее информационной модели.

Пусть, например, нас интересует, сколько времени будет падать парашютист с высоты 1000 м, если у него не раскроется парашют. Можно провести настоящий эксперимент — изготовить чучело парашютиста, сбросить его несколько раз с самолета и замерить среднее время падения. Можно, однако, вместо этого "поставить эксперимент на ЭВМ" — рассчитать время падения парашютиста, пользуясь информационной моделью. В этом параграфе мы покажем, как это делается.

Моделирование (***вычислительный эксперимент***) может оказаться незаменимым — ведь не всегда можно провести настоящий эксперимент. Мы не можем, например, устроить настоящую ядерную войну, чтобы узнать, как изменится климат, или лишить нашу планету озонового слоя, чтобы узнать, к чему это приведет.

Конечно, результаты вычислительного эксперимента могут оказаться и не соответствующими реальности, если в информационной модели будут не учтены какие-то важные стороны действительности. В задаче о падении парашютиста, например, для получения результата, соответствующего реальности, надо учитывать сопротивление воздуха, а при расчете запуска ракеты надо учитывать уже не только сопротивление, но и его изменение с ростом высоты.

26.2. МЕТОД ДИСКРЕТИЗАЦИИ НЕПРЕРЫВНЫХ ПРОЦЕССОВ

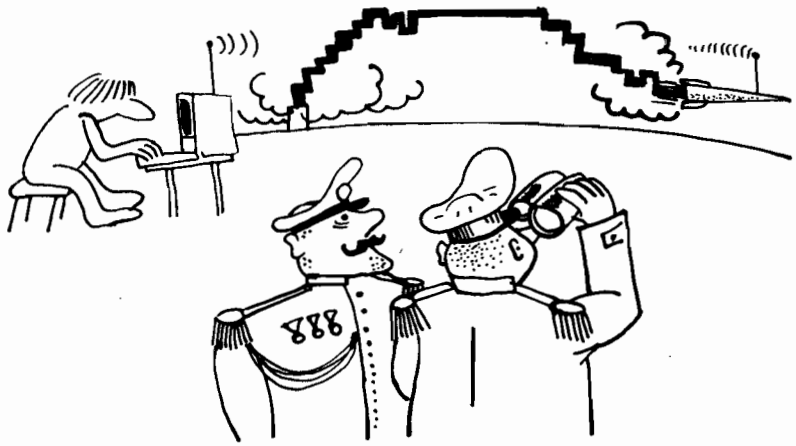
Вернемся к задаче падения парашютиста. Экспериментально установлено, что сила сопротивления воздуха пропорциональна квадрату скорости, а коэффициент зависит от формы тела. Поэтому ускорение падающего тела имеет вид $a=g-k\cdot v^2$, где k — коэффициент, зависящий от массы и формы тела (для человека среднего веса и роста $k\approx 0.004$).

Мы не знаем формул, выражающих время падения тела с таким ускорением (т. е. с учетом сопротивления воздуха). Как же вычислить это время? В таких случаях применяется ***метод дискретизации непрерывных процессов***. В чем он состоит?

Представьте, что мы засняли падение парашютиста на кинопленку. Кинопленка теперь — это некоторая модель реального процесса — по ней можно воссоздать падение, и даже достаточно точно. Реальный процесс падения, однако, является непрерывным, плавным. Кинопленка же состоит из отдельных кадров: на одном — парашютист в одном положении, на соседнем — в следующем, а между этими двумя положениями ничего нет. Подобного рода замена непрерывного процесса прерывным (принято говорить ***дискретным***) называется ***дискретизацией***.

При дискретизации время обычно разбивается на небольшие интервалы (например, по 0.01 с) и считается, что на протяжении одного интервала ничего не происходит ("ничего нет"). По истечении интервала "кадр" скачком меняется (например, уменьшается высота и увеличивается скорость парашютиста), потом опять ничего не происходит до окончания следующего интервала и т. д. Плавно уменьшающаяся высота парашютиста, например, при этом оказывается замененной на последовательность значений высот в моменты времени

$$t_1=0,\ t_2=0.01,\ t_3=0.02,\ \ldots$$



При моделировании таких дискретных процессов на ЭВМ информационная модель, как правило, описывает состояние процесса в один из моментов времени, а состояние в следующий момент определяется по рекуррентным соотношениям.

26.3. ПАДЕНИЕ С УЧЕТОМ СОПРОТИВЛЕНИЯ ВОЗДУХА

Итак, пусть требуется определить время падения парашютиста с учетом сопротивления воздуха. Для простоты будем считать, что падение происходит на планете с ускорением свободного падения $g=10.0$ (а не 9.81, как принято в физике).

Пусть для некоторого момента времени t нам известны высота тела h_t и скорость тела v_t. Тогда ускорение $a_t=g-kv_t^2$.

Подсчитаем высоту и скорость тела через промежуток времени dt:

$$h_{t+dt}=h_t-v_t*dt \text{ (считаем, что v не меняется);}$$
$$v_{t+dt}=v_t+a_t*dt \text{ (считаем, что a не меняется).}$$

Оба эти равенства тем точнее, чем меньше интервал времени dt, ведь чем меньше интервал, тем меньше меняются на нем скорость и ускорение. Полученные формулы и есть те рекуррентные соотношения, которые выражают следующее состояние процесса через предыдущее.

Используем для описания состояния падающего тела такую информационную модель:

```
вещ t | время, прошедшее с начала падения          (M19)
вещ h | расстояние до поверхности Земли
вещ v | скорость тела
```


Тогда рекуррентные соотношения можно переписать в виде:

```
a:=g-k*v**2
t:=t+dt
h:=h-v*dt
v:=v+a*dt
```

Вот какой алгоритм получается:

```
алг падение (арг вещ g, k, h0, v0, dt)                    (A120)
дано | g — ускорение свободного падения;
     | k — коэффициент сопротивления;
     | h0 — начальная высота, v0 — начальная скорость;
надо | напечатаны значения высоты и скорости в моменты време-
     | ни 0, dt, 2*dt, 3*dt и т. д. до момента удара о землю
нач вещ h, v, t, a
| t:=0; h:=h0; v:=v0
| вывод нс, "t=", t, "h=", h, "v=", v
| нц пока h>0
| | a:=g-k*v**2
| | t:=t+dt
| | h:=h-v*dt
| | v:=v+a*dt
| | вывод нс, "t=", t, "h=", h, "v=", v
| кц
кон
```

Приведем результаты выполнения этого алгоритма для $g=10$, $k=0.004$, $h0=1000$, $v0=0$, $dt=1$:

```
t=0     h=1000   v=0
t=1     h=1000   v=10.0
t=2     h= 990   v=19.6
          ...
t=22    h=  84   v=50.0
t=23    h=  34   v=50.0
t=24    h=-16    v=50.0
```

Последнее отрицательное значение высоты теоретически означает, что тело оказывается под землей на глубине 16 метров. Практически это значит, что на 24-й секунде падения тело ударится о землю и то, что с ним произойдет дальше, нашими формулами не описывается.

26.4. СРАВНЕНИЕ ПРИБЛИЖЕННОГО И ТОЧНОГО РЕШЕНИЙ

Если коэффициент сопротивления $k=0$ (например, при падении в вакууме), то положение и скорость тела в любой момент падения можно вычислить точно: $v(t)=v_0+gt$, $h(t)=h_0-v_0t-\frac{gt^2}{2}$.

В этом случае мы можем сравнить приближенное решение, которое дает алгоритм А126, с точным. Сделаем это для dt=1 и dt=0.5:

t	Шаг dt=1		Шаг dt=0.5		Точное решение	
	h	v	h	v	h	v
0	1000	0	1000	0	1000	0
1	1000	10	997.5	10	995	10
2	990	20	985	20	980	20
			...			
13	220	130	187.5	130	155	130
14	90	140	55	140	20	140
15	−50	150	−87.5	150	−125	150

Что можно сказать, глядя на эти числа?

1. Приближенное решение отличается от точного. Причину отличий легко увидеть: при dt=1 в первую секунду (от t=0 до t=1) тело вообще не падает. Это выглядит нелепо, но "что посеешь — то пожнешь" — ведь мы сами при подсчете высоты решили считать скорость неизменной на каждом промежутке времени, а в начале она равна нулю.

2. С уменьшением dt (при переходе от 1 к 0.5) приближенное решение приближается к точному.

3. Даже при dt=1 разница между приближенным и точным решением не так велика: оба они предсказывают, что тело ударится о землю на 15-й секунде падения.

26.5. ВЫБОР ШАГА ПО ВРЕМЕНИ

Обычно точный ответ приближенного вычисления неизвестен — ведь мы составляем алгоритм как раз для того, чтобы этот ответ узнать! Как же в этом случае оценить точность полученного решения? Исходя из каких соображений устанавливать величину dt?

Существуют строгие математические методы оценки погрешности вычислений, но мы рассмотрим только простейший прием, часто применяемый на практике. Будем уменьшать dt и смотреть, что происходит с ответом. Пусть, например, надо найти высоту тела через 20 секунд после начала падения (h=1000, v0=0, k=0.004). Вот значения h в момент t=20 при разных dt:

dt	1.0	0.5	0.2	0.1	0.05	0.001
h	184.0	178.3	175.2	174.2	173.7	173.3

Видно, что с уменьшением dt разница между соседними результатами становится все меньше (два последних отличаются

всего на полметра). Поэтому естественно предположить, что дальнейшее уменьшение dt уже не приведет к сильному изменению h, и взять dt=0.001.

Почему же сразу не взять очень маленькое dt, например dt=0.000001? Дело в том, что чем меньше dt, тем больше шагов придется сделать при выполнении алгоритма и тем больше времени уйдет на его выполнение. Если, например, при dt=0.001 алгоритм А126 выполняется за 1 с, то при dt=0.000001 он будет выполняться уже больше 15 мин. Другими словами, быстрее подсчитать 100 разных вариантов при dt>0.001, чем один при dt=0.000001.

Выбор dt, впрочем, обычно тоже поручают ЭВМ, т. е. составляют алгоритм, который начинает подсчет для какого-то значения dt, затем уменьшает dt (обычно вдвое), считает снова и т. д. до тех пор, пока результаты вычислений не станут достаточно близки друг к другу.

УПРАЖНЕНИЯ

1. Парашютист прыгнул с самолета, летящего со скоростью 180 км/ч на высоте 1300 м, и раскрыл парашют на высоте 600 м. Сопротивление воздуха пропорционально квадрату скорости ($k \approx 0.004$). Составьте алгоритм, который определяет время падения парашютиста до момента открытия парашюта.

2. В условиях упражнения 1 парашютист на высоте 1000 м группируется и коэффициент сопротивления уменьшается с 0.004 до 0.003. Составьте алгоритм, который вычисляет время падения парашютиста до момента открытия парашюта.

3. Составьте алгоритм, аналогичный алгоритму "падение", для расчета колебаний груза на пружинке (ускорение пропорционально величине отклонения от положения равновесия).

4. Шарик подвесили к пружине от школьного динамометра, оттянули вниз от положения равновесия на 1 см и отпустили. Жесткость пружины такова, что в момент отпускания шарика его ускорение под действием силы тяжести и силы упругости пружины равно -4 м/с^2. Составьте алгоритм, который определяет, через сколько секунд шарик поднимется на максимальную высоту.

5. Составьте алгоритм, который вычисляет координаты и скорость мяча, отпущенного на высоте h м над бесконечной наклонной плоскостью, наклоненной под углом α к горизонту, через t сек после начала движения. Удары упругие.

6. Решите упражнение 5, если при каждом отскоке мяча от плоскости модуль его скорости уменьшается на n%.

7. Тело движется по наклонной плоскости под действием силы тяжести. Сила сопротивления пропорциональна скорости тела. Составьте алгоритм, который вычисляет длину пути, пройденного телом за время t от начала движения.

8. На верхнюю ступеньку бесконечной лестницы (ширина ступенек 1, высота h) положили упругий мяч и покатили его со скоростью v. Считая мяч материальной точкой, а удары упругими, составьте алгоритм, определяющий номера первых n ступенек, о которые ударится мяч.

§ 27. КОМПЬЮТЕРНОЕ ПРОЕКТИРОВАНИЕ И ПРОИЗВОДСТВО

27.1. ЧЕРЧЕНИЕ НА ЭВМ

Что может дать ЭВМ конструктору и технологу на современном производстве? Прежде всего она может облегчить работу с чертежами. С помощью ЭВМ новый чертеж можно подготовить в несколько раз быстрее, чем на обычном кульмане. Если же чертеж уже хранится в ЭВМ и в него нужно внести небольшие изменения, то это можно сделать в десятки раз быстрее, чем за кульманом. Достаточно указать, какие части старого чертежа нужно заменить и что нужно поместить на их место, и ЭВМ создаст новый чертеж. Наиболее часто встречающиеся фрагменты чертежей, отдельные блоки и узлы можно хранить в памяти ЭВМ и использовать при создании новых чертежей. Использование такой библиотеки чертежей позволяет повысить производительность труда инженера за "электронным кульманом".

27.2. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Никакую новую конструкцию нельзя использовать без испытаний. Если при создании конструкции используется ЭВМ, то вся информация о конструкции есть в памяти ЭВМ. В этом случае испытания можно провести, не изготавливая конструкцию, а моделируя ее поведение на ЭВМ. При этом конструктор может вычислять различные характеристики (например, вес, объем, координаты центра тяжести), наблюдать работу конструкции в разных режимах (в том числе и в таких, какие невозможно или опасно воспроизвести на практике). Конструкцию можно легко менять в процессе этих компьютерных испытаний, выбирая наилучший вариант, изучать, как будут распределены напряжения при работе конструкции и пр. Подобное моделирование резко сокращает сроки разработки, позволяет повысить ее качество.

27.3. СТАНКИ С ЧИСЛОВЫМ ПРОГРАММНЫМ УПРАВЛЕНИЕМ (ЧПУ)

Если у Чертежника заменить бумагу листом металла, а перо резцом, то мы получим команды типа "опустить резцом", "сдвинуть резцом (**вещ** x, y)" и т. д. Такого рода устройства, работающие с реальными металлическими заготовками, называются станками ***с числовым программным управлением (ЧПУ)***. В состав станка

может входить и управляющая ЭВМ, в память которой по линиям связи поступает программа работы.

Меняя программу в памяти ЭВМ, можно перенастроить станок на производство нового типа деталей. Это позволяет создавать ***гибкие автоматизированные производства (ГАП)***, т. е. производства, перенастройка которых на выпуск другой продукции осуществляется сменой информации (программ) в памяти ЭВМ.

27.4. ПРОЕКТИРОВАНИЕ И ПРОИЗВОДСТВО — ЕДИНЫЙ ЦИКЛ

Рассчитав нужную деталь на ЭВМ и имея станки с числовым программным управлением, можно объединить проектирование и производство в единый цикл. При этом информация, полученная при проектировании, непосредственно, "не выходя из компьютера", будет использована для производства. Такой подход может значительно сократить сроки разработки и производства новых изделий. Имея в памяти ЭВМ требуемую форму детали, можно с помощью той же ЭВМ рассчитать, как должен двигаться резец станка, чтобы эту деталь изготовить. Зная траекторию резца, можно рассчитать скорость обработки, подачу охлаждающей жидкости и т. д. Использование ЭВМ позволяет изготавливать сложные детали безошибочно, с высокой точностью и без участия человека. Подачу заготовок со склада, перенос их от станка к станку и отправку на склад готовой продукции могут осуществлять управляемые ЭВМ роботы, транспортные тележки и пр.

Пусть требуется представить в ЭВМ поверхность сложной формы, например, капот автомобиля. Один из методов — так называемый ***метод конечных элементов*** — состоит в том, чтобы разбить поверхность капота на маленькие кусочки, которые приближенно можно считать плоскими, например на треугольники. Для задания такой составленной из треугольников поверхности в ЭВМ можно использовать информационную модель M20:

```
цел N                    | количество треугольников          (M20)
вещ таб c [1:N, 1:9]     | c [i, 1:9] — координаты вершин i-го
                         |              треугольника
```

УПРАЖНЕНИЯ

1. Модель M20 неэкономна: одна и та же вершина может входить в несколько треугольников и ее координаты будут храниться многократно. Измените модель M20 так, чтобы информация не дублировалась.

2. Считая, что толщина капота и плотность металла известны, составьте алгоритм подсчета веса капота в рамках а) модели М20; б) вашего решения упражнения 1.

3. Составьте информационную модель для представления объемных деталей и алгоритмы для нахождения а) веса; б) площади поверхности детали.

4. Придумайте способ задания температуры на поверхности модели М20. Составьте алгоритмы, вычисляющие: а) максимальную температуру модели; б) среднюю температуру модели; в*) площадь зоны поверхности, где температура выше 100°.

§ 28. ОТ ИНДУСТРИАЛЬНОГО ОБЩЕСТВА К ИНФОРМАЦИОННОМУ [ЗАКЛЮЧЕНИЕ]

Мы рассмотрели лишь некоторые из наиболее крупных областей применения ЭВМ. Перечислить их все в настоящее время уже вряд ли возможно — счет персональным, домашним, игровым, встроенным и иным ЭВМ уже пошел на сотни миллионов. ЭВМ встраиваются в самолеты и автомобили, в часы, стиральные машины, кухонные комбайны и даже в спортивную обувь. Использование ЭВМ позволило, например, создать систему спутниковой навигации автомобилей (когда на экране перед водителем, где бы он ни оказался, изображается карта окружающей местности и точное положение автомобиля). Применение ЭВМ открыло путь к "всемирной библиотеке" — возможности, не выходя из дома, получить копию любой книги, статьи, описание того или иного изобретения и т. п. В развитых странах человек со своего домашнего компьютера может заказать билеты на поезда, самолеты, корабли по сложному маршруту со многими пересадками, забронировать на нужные числа места в гостиницах и даже заказать билеты в театр в пунктах пересадки. И это только начало становления глобальных информационных сетей!

Но путешествуем мы не каждый день, а вот покупаем что-нибудь почти ежедневно. И здесь ЭВМ тоже может помочь.

28.1. ЭЛЕКТРОННЫЙ МАГАЗИН, ШТРИХОВОЙ КОД И ЭЛЕКТРОННЫЕ ДЕНЬГИ

Быть может, вы встречали на некоторых импортных товарах прямоугольник из черных и белых полосок (фото вклейки). Это уникальный ***штриховой код*** товара. В современных магазинах кассир не должен ни набирать стоимость товара, ни даже помнить ее (да это и невозможно, когда в одном магазине насчитывается свыше 30 тыс. наименований разных товаров). Достаточно провести штриховой код мимо считывающего устройства кассового аппарата, и ЭВМ сама определит цену товара, а в конце изобразит на табло стоимость всех покупок.

Про каждый товар ЭВМ магазина помнит не только его текущую цену (а цена может меняться в зависимости от того, каким спросом пользуется товар), но и его количество. Если запасы каких-то товаров на исходе, то ЭВМ сама (по информационной сети) пошлет запрос на склад. ЭВМ склада, получив такие запросы от разных магазинов, спланирует оптимальную загрузку транспорта, маршруты перевозок — и к утру все товары будут на местах.

А что же наш покупатель? Ведь ему надо рассчитаться за товар. Не думайте, что он станет считать бумажки и пересчитывать сдачу. Для расчетов используются ***электронные деньги*** — специальные пластиковые карточки, особым образом хранящие информацию о банковском счете покупателя. Достаточно вставить эту карточку в кассовый аппарат — и ЭВМ сама перечислит нужную сумму со счета покупателя на счет магазина (точнее, пошлет запрос в банк, а уж ЭВМ банка произведет нужные перечисления). При такой методике на обслуживание одного покупателя кассир тратит секунды, а очередей просто не бывает.

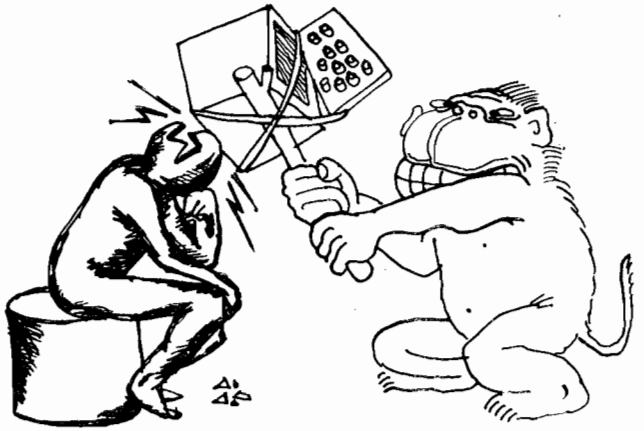
28.2. ПРОНИКНОВЕНИЕ ЭВМ ВО ВСЕ СФЕРЫ ЖИЗНИ

Компьютер можно использовать не только для работы, но и для отдыха. Появление компьютеров привело и к революции в области игр. Число компьютерных игр, появившихся за последние два десятилетия, уже превзошло число игр, изобретенных человечеством за всю предшествующую историю цивилизации. Значительная доля существующих в мире компьютеров используется для игр.

Компьютеры вторгаются во все сферы жизни. Появились даже компьютерные преступления (когда, например, программа начисления заработной платы переводит незаработанные деньги на счет автора программы). Другой пример: несколько лет назад один из программистов ВАЗа в знак протеста против низкой зарплаты внес умышленную ошибку в программу и этим остановил на несколько дней главный конвейер (вспомните п. 23.2). В результате завод понес большой материальный ущерб, не сопоставимый с зарплатой всех программистов ВАЗа, вместе взятых, а программист был дисквалифицирован и переведен в рабочие.

28.3. ОШИБКИ В ПРИМЕНЕНИЯХ ЭВМ

Мы много говорили о достоинствах ЭВМ и об их роли в жизни общества. Однако, как и любое другое изобретение человека, компьютер может принести не только пользу, но и вред. Представление о том, когда ЭВМ использовать нецелесообразно, каковы основные ошибки в их применениях, является важной частью компьютерной грамотности. Поэтому мы кратко перечислим несколько таких случаев.



1. ***Превращение ЭВМ из средства в цель.*** Применение ЭВМ само по себе отнюдь не служит признаком технического прогресса. Скорее наоборот — прогресс чаще оказывается связан не с усовершенствованием существующей, а с переходом на новую технологию. Например, переход на точное литье упраздняет чистовую механическую обработку деталей и делает ненужной ЭВМ, управляющую этой обработкой. Стремление "внедрить ЭВМ" может воспрепятствовать такому переходу и тем самым затормозить научно-технический прогресс.

Аналогично, отмена дополнительной платы за междугородные телефонные разговоры может сделать ненужной ЭВМ, вычисляющую их стоимость в зависимости от длительности разговора и расстояния между городами. Строительство туннелей и эстакад может упразднить светофоры и регулирование движения с помощью ЭВМ. Переход к новым принципам оплаты труда, налогового обложения и социального обеспечения может сделать ненужным расчет зарплаты на ЭВМ и т. п.

2. ***Ошибки в алгоритмах.*** ЭВМ лишь выполняет алгоритмы. Эти алгоритмы могут быть составлены с ошибками или на основе неверных представлений о действительности. Например, одна из первых компьютерных систем противовоздушной обороны США (60-е годы) в первое же дежурство подняла тревогу, приняв восходящую из-за горизонта Луну за вражескую ракету, поскольку этот "объект" приближался к территории США и не подавал сигналов, что он "свой".

3. ***Неверные исходные данные.*** Результат работы ЭВМ зависит не только от алгоритма, но и от обрабатываемой информации. Ошибки в исходных данных не менее опасны, чем ошибки в алгоритмах. Несколько лет назад, например, в Антарктиде разбился самолет с туристами на борту, поскольку в управляющую полетом ЭВМ были помещены неверные координаты аэропорта взлета и ЭВМ ошибочно рассчитала высоту полета над горами.

4. ***ЭВМ не всемогущи.*** Далеко не всякая задача обработки информации может быть решена с помощью ЭВМ. Существуют задачи, алгоритмы решения которых в настоящее время неизвестны. Например, до сих пор не существует приемлемых алгоритмов, которые позволили бы отличить на фотографии кошку от собаки или грамотно перевести художественное произведение с одного языка на другой. Бывает и такое, что алгоритм известен, но выполнить его нельзя, так как даже самым быстродействующим ЭВМ для его выполнения понадобятся миллионы лет (пример такой задачи — безошибочная игра в шахматы). Поэтому глубоко ошибочно представление о том, что если человек не знает решения задачи, то ее надо "заложить в ЭВМ" и ЭВМ даст ответ.

5. ***Недооценка социальных последствий компьютеризации.*** Наконец, и это самое важное, использование ЭВМ меняет жизнь людей. Поэтому вопрос о новых применениях ЭВМ прежде всего должен рассматриваться с точки зрения социальных последствий, а не с позиции "могут это ЭВМ" или "не могут", выгодно это или не выгодно. Многие этапы информатизации общества имеют трудно предсказуемые социальные последствия. Внедрение заводов-автоматов требует перевода значительной части работающих из производственной сферы в сферу обслуживания. Если работа в сфере обслуживания считается в обществе менее престижной, такой перевод может вызвать социальную напряженность. Организация работы на дому позволяет увеличить количество свободного времени, но разрушает сферу общения с сослуживцами. Распространение компьютерных игр приводит к тому, что дети быстрее развиваются, но меньше бывают на воздухе и меньше общаются друг с другом. Во многих случаях ЭВМ просто не следует внедрять. Например, не следует поручать ЭВМ человеческих дел, связанных с принятием моральных и этических решений при воспитании детей, формулировании целей социального развития общества, установлении виновности обвиняемых в преступлении.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА ДЛЯ ЧТЕНИЯ

1. Знакомьтесь: компьютер.— М.: Мир, 1989.

2. Вейценбаум Дж. Возможности вычислительных машин и человеческий разум.— М.: Радио и связь, 1982.