

А. И. Стрелянов

ПРОИЗВОДСТВО
ВЫЧИСЛЕНИЙ
НА
ПРОГРАММИРУЕМЫХ
МИКРО-
КАЛЬКУЛЯТОРАХ



МАШИНОСТРОЕНИЕ

А. И. Стрелянов

ПРОИЗВОДСТВО
ВЫЧИСЛЕНИЙ
НА ПРОГРАММИРУЕМЫХ
МИКРОКАЛЬКУЛЯТОРАХ
(МК-52, МК-54, МК-61)



Ленинград
«Машиностроение»
Ленинградское отделение
1990

В настоящее время для широкого круга лиц, не являющихся специалистами в области информатики и вычислительной техники, встала задача не столько теоретического, сколько практического использования в своей работе индивидуальных электронно-вычислительных средств. Среди последних наиболее доступными являются программируемые микрокалькуляторы (ПМК), так как они обладают низкой стоимостью и высокой надежностью, компактностью, возможностью переноски и независимостью от внешних источников питания. Все это дает возможность пользователю иметь ПМК всегда при себе постоянно готовыми к работе, поэтому при решении относительно простых задач ПМК превосходят более мощные средства вычислительной техники. Большинство вычислителей-непрофессионалов значительную часть времени затрачивает на решение именно таких задач. Все это говорит о том, что ПМК следует рассматривать не как временное явление, обусловленное недостаточным распространением персональных компьютеров, а как перспективные вычислительные средства индивидуального пользования для вполне определенного и достаточно широкого круга задач.

С 1986 г. нашей промышленностью освоено массовое производство ПМК типа «Электроника»: МК-61, -52. Последний отличается от первого помимо внешнего оформления наличием дополнительной памяти для хранения ранее отложенных программ в выключенном состоянии ПМК. В эксплуатации находится также большое количество выпущенных до 1986 г. ПМК «Электроника»: Б3-34 и МК-54, отличающихся друг от друга типом источника питания (в первом используются аккумуляторы, а во втором — гальванические элементы А-316). Все перечисленные модели ПМК являются программно-совместимыми снизу вверх, т. е. программы, составленные для МК-54 (Б3-34), будут пригодны и для МК-61, -52 (но не наоборот). Следует подчеркнуть, что все перечисленные модели ПМК предназначены только для индивидуального пользования. В качестве временной меры, обусловленной недостаточным распространением персональных компьютеров, для организаций был выпущен в больших количествах программируемый микрокалькулятор МК-56, представляющий собой настольный вариант МК-54 с питанием только от сети.

Для пользователей-непрофессионалов уже выпущена обширная литература по ПМК, посвященная различным аспектам их применения и программирования [2, 11—13, 16—17, 19—21, 23]. Между тем, для широкого круга пользователей необходима не только эта литература, но и практическое руководство по самостоятельному обучению производству вычислений на ПМК. Предлагаемая книга представляет собой попытку создать такое руководство.

Стрелянов А. И.

С 84 Производство вычислений на программируемых микрокалькуляторах (МК-52, МК-54, МК-61). — Л.: Машиностроение. Ленингр. отд-ние, 1990. — 272 с.

ISBN 5-217-00751-6

Книга охватывает все основные вопросы производства вычислений на программируемых микрокалькуляторах (ПМК): вычисления в ручном режиме, разработку алгоритмов и программ, их отладку, выполнение вычислений по отложенным программам, разработку программных комплексов. Подробно изложены принципы и способы решения на ПМК задач методом статистических испытаний, приведены примеры построения программных комплексов, реализующих на ПМК статистические модели. В качестве примеров рассмотрено большое количество алгоритмов (описанных на алгоритмическом языке: изучаемом в курсе информатики средней школы) и соответствующих им программ для ПМК. Дано большое количество упражнений с ответами и решениями.

Для широкого круга ИТР, студентов, не являющихся специалистами в области информатики и вычислительной техники, желающих самостоятельно освоить и использовать в своей работе ПМК.

с 2404060000—998
038 (01)—90 КБ 34-25—89

ББК 32.973

ISBN 5-217-00751-6

© А. И. Стрелянов, 1990

В качестве базовой модели рассматривается наиболее массовая модель «Электроника МК-61». Однако книга может быть использована и для обучения вычислениям на ПМК «Электроника»: МК-52, -54. Все, что излагается по МК-54, относится и к другим вариантам его оформления (Б3-34 и МК-56). Ограничения на использование МК-54 по сравнению с МК-61, а также дополнительные возможности МК-52 указываются в тексте книги по мере необходимости. При чтении книги необходимо иметь под рукой программируемый микрокалькулятор (любой из перечисленных моделей), чтобы сопровождать на нем все описываемые действия и сопоставлять полученные результаты с приведенными в книге.

Любой ПМК является простейшей электронно-вычислительной машиной, включающей в себя все основные ее элементы, а именно: процессор, автоматически выполняющий введенную в память программу и обрабатывающий данные; память, способную хранить программу и данные; устройства ввода и вывода (клавиатуру и индикатор). Поэтому знания и навыки, полученные в процессе освоения вычислений на ПМК, могут быть использованы в дальнейшем при освоении вычислений на персональных компьютерах. В книге рассматриваются основные сведения, знание которых необходимо для практического использования ЭВМ всех типов: разработка алгоритмов и программ, их тестирование и отладка, разработка программных комплексов из ранее составленных программ, решение задач по отложенным программам. Впервые показано, как разработать и реализовать на ПМК программные комплексы для решения задач методом статистических испытаний. Рассмотрены конкретные примеры реализации статистических моделей на ПМК.

Обучение программированию производится на основе алгоритмического языка (лексикона), предложенного академиком А. П. Ершовым и изучаемого в средней школе в рамках предмета «Информатика и вычислительная техника» [14]. Алгоритм решения задачи, первоначально разработанный и записанный на лексиконе, уточняется на втором уровне представления БЕЙСИК-командами, а последние уточняются командами программы ПМК. Весь процесс преобразования исходного алгоритма на лексиконе в БЕЙСИК-алгоритм и ПМК-программу формализован с помощью шаблонов преобразования. Используемый способ преобразования исходного алгоритма в команды программы [22] может быть применен и для персональных компьютеров, входным языком которых является язык БЕЙСИК (в отличие от ПМК здесь будет достаточно двух уровней представления — на лексиконе и на языке БЕЙСИК). Все это позволяет доводить до числовых результатов на ПМК или ЭВМ алгоритмы, рассматриваемые при изучении указанного предмета средней школы.

Автор благодарит кандидата технических наук В. А. Хмелюк за помощь, оказанную при подготовке издания.

Раздел I

ОБЩИЕ СВЕДЕНИЯ

О ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРАХ

И ВЫПОЛНЯЕМЫХ ИМИ

ВЫЧИСЛИТЕЛЬНЫХ ОПЕРАЦИЯХ

Глава 1. ПРИНЦИПЫ ПОСТРОЕНИЯ И ФУНКЦИОНИРОВАНИЯ ПМК

Термин «микрокалькулятор» закреплен Общесоюзным стандартом за переносными электронными вычислительными устройствами с автономным питанием карманного размера и небольшой массы (50—400 г). Исходя из этого микрокалькуляторы должны удовлетворять в первую очередь требованиям минимальной массы, независимости от внешних источников питания, высокой надежности в работе при удовлетворительной точности вычислений (8—10 значащих цифр). По своим функциональным возможностям все выпускаемые в настоящее время микрокалькуляторы можно подразделить на две группы: непрограммируемые (НМК) и программируемые (ПМК).

Непрограммируемыми называются микрокалькуляторы, выполняющие операции над числами только по командам, подаваемым нажатием соответствующих клавиш. Простейшие из них выполняют лишь арифметические операции; микрокалькуляторы, предназначенные для инженерных расчетов (инженерные МК), вычисляют также наиболее часто встречающиеся при расчетах функции (например, МК-51).

Программируемыми микрокалькуляторами (ПМК) называются микрокалькуляторы, выполняющие вычисления как в обычном (ручном) режиме, так и в автоматическом (программируемом) режиме. В ручном режиме вычисления на ПМК производятся так же, как и на непрограммируемых микрокалькуляторах: ввод команд производится нажатием соответствующих клавиш, а выполнение — немедленно после их нажатия. В автоматическом режиме вычисления производятся по программе, предварительно вводимой в запоминающее устройство ПМК, называемое программной памятью.

В процессе выполнения расчетов на ПМК пользователю приходится оперировать различными клавишами на его пульте управления. При этом ему необходимо знать, что должно произойти после нажатия каждой клавиши, т. е. какие операции будут выполнены и что в результате будет отображено на экране. А для этого необходимо иметь некоторое представление об устройстве ПМК, характеристиках его элементов и их взаимодействии в про-

цессе работы. Эти вопросы в объеме, необходимом для грамотного выполнения пользователем вычислений на ПМК, будут рассмотрены в настоящей главе.

1.1. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ И ХАРАКТЕРИСТИКИ ПМК

Требования, определяющие назначение вычислительных устройств данного класса. К числу таких требований относятся:

портативность, определяемая массой и габаритами, позволяющими переносить ПМК в кармане одежды, сумке, портфеле. Это требование легко выполняется во всех выпускаемых ПМК, масса которых не превышает 250 г, а габариты — 80×180×38 мм. Такие габариты являются, по-видимому, оптимальными, так как дальнейшее их уменьшение делает клавиатуру ПМК неудобной для пользователя;

автономность электропитания, характеризуемая длительностью непрерывной работы ПМК от внутренних автономных источников питания (гальванических элементов или аккумуляторов) без их подзарядки от сети. Время непрерывной работы выпускаемых ПМК от автономного источника питания составляет не более 6 ч, что явно недостаточно. Поскольку из-за требования портативности не представляется возможным увеличить емкость источников питания, то время непрерывной работы определяется главным образом мощностью, потребляемой ПМК при выполнении вычислений. Эта мощность (0,4—0,7 Вт) является значительной для применяемых источников питания (4 элемента А-316). Поэтому все ПМК комплектуются отдельными выпрямителями для питания от сети;

низкая стоимость, делающая ПМК доступным для широкого круга пользователей (студентов, инженеров, научных работников). Она должна быть по крайней мере на порядок меньше стоимости простейшей микроЭВМ. Это требование близко к выполнению. Розничная цена ПМК находится в пределах 65—115 руб. Цена простейших микроЭВМ 500—1000 руб.

Требования, обеспечивающие простоту и удобство пользования. Эти требования обусловлены тем, что ПМК эксплуатируется и обслуживается одним пользователем, не являющимся, как правило, специалистом в области вычислительной техники и программирования на ЭВМ. К числу данных требований относятся:

надежность и постоянная готовность к вычислениям (вручную и/или по заранее составленной и введенной в память ПМК программе), обеспечивающие кроме других мер ограничением быстродействия до приемлемого уровня в 2 оп.с;

простота и удобство вычислений, выполняемые за счет использования языка клавишного набора в качестве средства общения ПМК с пользователем. Этот язык является языком иероглифов. Каждый иероглиф обозначает клавишу и определяет операцию, предписываемую к выполнению после ее нажатия. Програм-

ма решения задачи представляется в этом случае последовательностью иероглифов (обозначений клавиш), предписывающих выполнение соответствующей последовательности действий. Использование такого языка (по сравнению с набором предписаний словами, составленными из символов алфавита, как это принято в языках программирования для ЭВМ) резко сокращает длину программ и число ручных операций, выполняемых пользователем при вычислениях. Вместе с тем это определяет значительно большее разнообразие и количество типов команд по сравнению с ЭВМ. Поэтому число наименований различных команд является одной из важных характеристик ПМК;

совместимость языка вновь выпускаемых моделей ПМК с ранее выпущенными, означающая, что программа, составленная для ранее выпущенных и находящихся в эксплуатации моделей ПМК, должна выполняться и на выпущенных позднее и вновь разрабатываемых моделях (но не наоборот!). Это требование выполняется для всех моделей ПМК, кроме первой, морально устаревшей модели ПМК «Электроника Б3-21»;

логическая организация ПМК, под которой понимаются правила ввода данных и последовательность выполнения операций над ними. По способу логической организации различают ПМК с алгебраической логикой и бесскобочной логикой. В первых арифметические выражения вводятся в ПМК посимвольно (включая скобки в том же порядке, в котором они обычно записываются на бумаге), а после набора знака равенства — вычисляются. Во вторых для вычисления выражения необходимо привести его к бесскобочному виду, который принято называть польской инверсной записью (подробнее о ней см. в следующей главе). Данные вводятся в порядке, предусмотренном этой записью, а вычисления производятся после ввода знака операции. При этом в качестве операнда используется результат предыдущей операции. Для составления программ более удобны ПМК с алгебраической логикой, однако при этом существенно усложняется внутренняя структура ПМК и, кроме того, удлиняются программы решения задач (за счет необходимости ввода скобок и знака равенства), что требует увеличения емкости программной памяти ПМК. В то же время, как показывает опыт, пользователь ПМК легко привыкает к бесскобочной логике. Поэтому в настоящее время все выпускаемые ПМК рассчитаны на использование бесскобочной логики;

удобство ввода и отладки программ, состоящее в том, чтобы программа вводилась в память ПМК только один раз и хранилась там неопределенно долго (пока не будет стерта пользователем за ненадобностью). К сожалению, это требование в большинстве ПМК (за исключением МК-52) не выполняется. Введенная в память ПМК программа при выключении ПМК стирается, поэтому основным способом ввода ее в ПМК является ручной набор на пульте непосредственно перед выполнением. По этой причине

Основные технические характеристики ПМК

Характеристики	Модель ПМК				
	Б3-34, 1980 г.	МК-54, 1982 г.	МК-56, 1982 г.	МК-61, 1985 г.	МК-52, 1986 г.
Масса, г	390	250	1300	250	250
Габаритные размеры, мм	185×100×48	167×78×38	208×205×60	167×78×38	80×180×38
Источник питания	Аккумуляторы Д-0,5×4, сеть	Элементы А-316×4, сеть	—	Элементы А-316×4, сеть	Элементы А-316×4, сеть
Время непрерывной работы от автономного источника, ч	3	6	—	6	6
Розничная цена, руб.	85	65	160	85	115
Количество различных кодов команд	188	188	188	211	211
Выполняемые команды:					
+ — × : 1/x, \sqrt{x} , x^2 , e^x , 10^x , $\ln x$, $\lg x$, X^y , $\sin x$, $\cos x$, $\tg x$, $\arcsin x$, $\arccos x$, $\arctg x$, π	есть	есть	есть	есть	есть
x , {x}, $\max(x, y)$, \vee , \wedge , \oplus , ИНВ, СЧ	нет	нет	нет	есть	есть
Прямой и обратный перевод значений: угла: градусы — минуты, \leftrightarrow градусы и доли градуса; времени: часы — минуты — секунды \leftrightarrow часы и доли часа	нет	нет	нет	есть	есть
Число разрядов мантиссы/порядка	8/2	8/2	8/2	8/2	8/2
Быстродействие арифметических оп./с	2	2	2	2	2
Емкость памяти: ОЗУ данных, регистров стековой, адресуемой;	4 14	4 14	4 14	4 15	4 15
ОЗУ программ, шагов	98	98	98	105	105
ППЗУ, байт	нет	нет	нет	нет	512
Интерфейс для подключения внешних устройств	нет	нет	нет	нет	есть

максимальная длина программы, а следовательно, и емкость программной памяти, ограничивается длиной в 98—105 шагов.

Удобство ввода программы зависит также от способа отображения команд программы на экране. Оно должно совпадать с обозначениями клавиш, предписывающих выполнение соответствующих команд. Во всех выпускаемых ПМК это требование не выполняется: на экране отображаются шестнадцатеричные коды команд, не совпадающие с обозначениями клавиш. При этом отображения шестнадцатеричных цифр не полностью соответствуют их общепринятым обозначениям. Причина невыполнения этого требования обусловлена трудностью отображения символов-иероглифов на экране индикатора, особенно если он рассчитан на отображение только числовых данных.

Для того чтобы обеспечить выполнение всех перечисленных требований первой и второй группы, ПМК предназначают только для обработки числовых данных в показательной форме.

В какой степени существующие ПМК отвечают перечисленным требованиям, показано в табл. 1.1.

Как уже отмечалось, все ПМК рассчитаны на выполнение вычислений в двух режимах — ручном и автоматическом. Необходимость их использования в ПМК обусловлена следующим. Для работы в автоматическом режиме необходимо вручную набрать программу на пульте ПМК. А это требует практически столько же времени, как и решение задачи в ручном режиме. Если же учесть, что время составления и отладки программы измеряется в часах (а сам счет в минутах), то составление программы для проведения расчета (пусть и трудоемкого, но единственного) вряд ли целесообразно. В таком случае целесообразно использовать ПМК в ручном режиме. К этому же режиму приходится прибегать и в том случае, когда программа не помещается в программную память. Поэтому пользователь ПМК должен овладеть практическими навыками решения задач не только в программируемом, но и в ручном режиме.

1.2. УСТРОЙСТВО ПМК

Любой программируемый микрокалькулятор можно рассматривать как разновидность ЭВМ, предназначенной для автоматического выполнения операций только над числовыми данными. В этом смысле ПМК не отличается от первых ЭВМ, которые не могли обрабатывать текст. Как и всякая ЭВМ, ПМК состоит из следующих основных элементов:

устройств ввода-вывода данных (пульт управления и индикатор);

памяти, предназначенней для хранения программ и данных (ОЗУ, ПЗУ, ППЗУ);

процессора, предназначенного для выполнения вычислений и управления всем ходом вычислительного процесса по введенной в память программе (рис. 1.1).

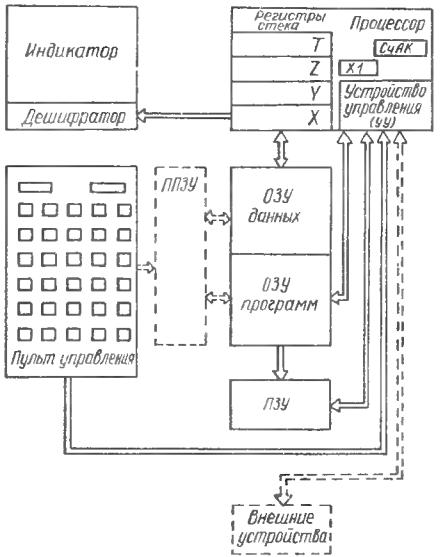


Рис. 1.1. Упрощенная схема ПМК



Рис. 1.2. Внешний вид МК-61

1.2.1. УСТРОЙСТВА ВВОДА—ВЫВОДА

Устройствами ввода—вывода для программируемого микрокалькулятора служат:

пульт управления, предназначенный для ввода числовых исходных данных и программ пользователя;

индикатор, предназначенный для отображения на экране вводимых исходных данных и результатов вычислений, а также кодов команд программы при ее вводе.

Пульт управления предназначен для ввода исходных данных и программ, а также для ввода в процессе вычислений управляющих воздействий пользователя. В качестве примера рассмотрим пульт управления ПМК «Электроника МК-61» — базовой модели ПМК (рис. 1.2). Сверху под индикатором на пульте расположены два переключателя: слева — выключатель питания (положение вправо — включено), справа — переключатель Р — ГРД — Г (радианы — градусы), соответствующие положения которых предписывают ПМК выполнять вычисления либо в радианах, либо в градусах (сотых долях прямого угла), либо в градусах.

Ниже указанных переключателей расположена клавиатура пульта управления, состоящая из 30 клавиш (6 рядов по 5 клавиш в каждом), с помощью которых может быть набрано 212 различных команд (189 для МК-54). В первом приближении под словом «команда» будем понимать элементарное действие, выполняемое ПМК в результате нажатия соответствующей клавиши на пульте управления.

Каждая клавиша, вообще говоря, имеет не одно, а два или три обозначения и выполняет функции не одной, а двух—трех клавиш. Основное назначение клавиши указано символами белого цвета на самих клавишиах. Так обозначены, например, клавиши набора чисел и клавиши, предписывающие выполнение основных арифметических операций:

0 ... 9 и **+** **-** **×** **÷**.

Желтым и голубым цветами над клавишами даны другие обозначения. Для выполнения команд, соответствующих желтым обозначениям, необходимо предварительно нажать префиксную клавишу с черным символом **F** на желтом фоне. Например, для возведения ранее набранного и отображаемого на индикаторе числа в квадрат необходимо нажать сначала префиксную клавишу **F**, а затем

клавишу, над которой желтым цветом обозначено **x²** (основное назначение этой клавиши — умножение **×**). Для выполнения команд, соответствующих голубым обозначениям, необходимо предварительно нажать префиксную клавишу с белым символом **K** на голубом фоне. Например, для перевода угловых величин, выраженных в градусах, минутах и долях минут, отображаемых на экране индикатора, в значения, выраженные в градусах и долях градуса, необходимо предварительно нажать на префиксную клавишу **K**, а затем на клавишу, над которой голубым цветом

справа помещены символы **o°** (основное назначение этой клавиши — сложение, символ на клавише — знак **+**). В зависимости от используемого обозначения нажатие клавиши будет предписывать выполнение той или иной команды.

Обозначения префиксных клавиш **F** или **K** условимся помещать впереди основных обозначений клавиш (можно без интервала). Например, **F10^x**, **Fe^x**; **KСЧ**, **КНОП**. Для упрощения обозначения клавиш не будем также помещать их в прямоугольники.

Обычно нажатию одной клавиши (не считая нажатия префиксной) соответствует одна команда, обозначенная символом на клавише (над клавишей или под клавишей). Однако часть команд, как будет показано ниже, реализуется путем нажатия двух клавиш (не считая префиксной).

Все клавиши, составляющие клавиатуру пульта управления, можно условно подразделить на 6 групп.

1. Клавиши набора чисел:

десять клавиш для набора цифр 0 ... 9;
ввода знака отрицательного числа /—/;
ввода десятичной точки .;

броса неправильно введенного числа Сх;
разделения последовательно вводимых чисел В↑.

2. Клавиши, предписывающие выполнение одноместных операций (т. е. выполняемых над одним числом, отображаемым на экране индикатора). Это, главным образом, клавиши, предписывающие вычисление стандартных функций одного аргумента. Их назначение указано сверху над клавишами желтым или голубым цветом. Например, sin — желтого цвета, [x] (выделение целой части числа) — голубого цвета. Перед нажатием каждой из указанных клавиш всегда нажимается префиксная клавиша F или K. После набора числа (ввода аргумента) и нажатия клавиши с обозначением соответствующей функции ее значение отображается на экране индикатора.

3. Клавиши двухместных операций, выполняемых над двумя числами. К ним относятся: + — × ÷ X_у K_{max}. После ввода в ПМК двух чисел и нажатия какой-либо из перечисленных клавиш выполняется двухместная операция и результат ее отображается на экране индикатора.

4. Клавиши обращения к памяти. Это две рядом расположенные клавиши: x → П (запись числа в память) и П → x (чтение ранее записанного числа из памяти). Команды x → П и П → x выполняются после последовательного нажатия двух клавиш: сначала клавиши x → П или П → x, а затем клавиши, указывающей номер (адрес) регистра памяти, к которому следует обратиться для записи или чтения числа.

5. Клавиши управления вычислениями:

перемещения чисел в операционных регистрах (т. е. в регистрах, в которых помещаются числа-операнды для выполнения

над ними арифметических операций): ↔ и ↴ :

восстановления результата предыдущего действия Вх.

6. Клавиши управления работой программы:

ФАВТ — перевод ПМК в режим вычислений;

ФПРГ — перевод ПМК в режим ввода программы в программную память;

С/П — стоп/пуск, предписывающий запуск программы, хранящейся в программной памяти ПМК, или останов выполняемой программы;

В/0 — возврат к нулю счетчика адреса команд после выполнения программы или возврат в основную программу после выполнения подпрограммы;

и ряд других клавиш, подробное назначение которых будет дано ниже (по мере изложения материала по их практическому использованию).

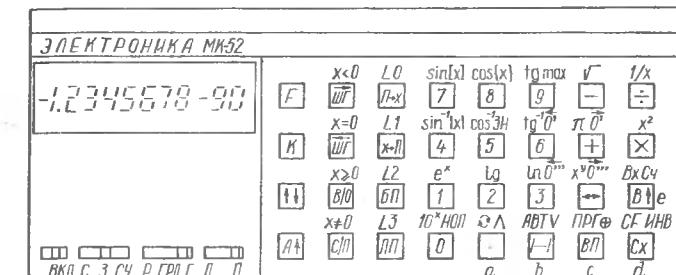


Рис. 1.3. Внешний вид МК-52

Программируемый микрокалькулятор «Электроника МК-52» имеет другое внешнее оформление — горизонтальное. Справа от экрана индикатора размещается 4 ряда клавиш по 8 в каждом (рис. 1.3), всего 32 клавиши. Две дополнительные клавиши предназначены для команд обмена данными и программами записи — считывания с полупостоянным запоминающим устройством ППЗУ, которое служит для хранения программ и данных при выключенном состоянии ПМК. Обозначения и выполняемые команды на остальных 30 клавишах полностью совпадают с таковыми на МК-61. В отличие от МК-61 выключатель ВКЛ и переключатель Р—ГРД—Г расположены слева под экраном индикатора. Там же помещены дополнительные переключатели: С—З—СЧ (стирание—запись—считывание из ППЗУ или в ППЗУ) и Д—П (данных или программ).

Программируемый микрокалькулятор МК-54 по внешнему виду не отличается от МК-61. Единственное их различие заключается в том, что в МК-54 клавиши имеют в основном только два обозначения: на самих клавишах (белого цвета) и над клавишами (желтого цвета). Обозначения голубого цвета, как в МК-61, отсутствуют. Не предусмотрены в МК-54 и соответствующие команды. Во всем остальном обозначения клавиш МК-54 и МК-61 и выполняемые при их нажатии команды совпадают.

Расположение клавиш ПМК «Электроника Б3-34» и все выполняемые команды полностью совпадают с таковыми в МК-54. Однако в Б3-34 принято другое обозначение клавищ. Соответствие между обозначениями клавиш МК-54 и Б3-34 дано в прил. 1.

Следует отметить, что обозначения клавиш ПМК, удобные для выполнения вычислений вручную, не всегда удобны для записи программ и их тиражирования в печатных изданиях. Поэтому все авторы сборников программ для ПМК делают попытки упрощения записи обозначений клавиш (команд) в машинописном или печатном текстах [2, 11—13, 16, 17, 20, 21]. Общепринятым упрощением является отказ от помещения обозначений клавиш в прямоугольники. Наряду с этим используются такие упрощения, как запись обозначения на одной строке (как это принято в языках програм-

мирования для всех типов ЭВМ), замена символов на сходные, но имеющиеся на клавиатуре пишущей машинки, сокращение числа символов в обозначении клавиши. В данном пособии также будут использоваться упрощенные в написании и несколько измененные названия команд ПМК по сравнению с их обозначениями на пульте управления. Все эти изменения будут оговорены и при необходимости обоснованы по ходу изложения материала.

Индикатор (люминесцентный) предназначен для отображения: десятичных чисел (исходных данных), набираемых на клавиатуре пульта управления;

результатов вычислений в виде десятичных чисел; кодов и адресов команд при вводе программы.

Индикатор ПМК содержит 12 знакомест, на каждом из которых может быть отображен один из следующих символов: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, —, L, Г, І, Е. Кроме того, дополнительно к основному символу на каждом из знакомест может быть отображен внизу справа десятичная точка. Цифры от 0 до 9, десятичная точка и знак минус /—/ используются для отображения на индикаторе чисел, а символы L, Г, І, Е наряду с цифрами и знаком минус — для отображения кодов команд и сигнала ошибки ЕГГОГ при выполнении некорректных операций.

Десятичные числа могут отображаться на экране индикатора как в естественной форме (в виде целого числа или неправильной десятичной дроби), так и в показательной форме с отдельным отображением на экране (в левой части) мантиссы числа, а в правой части — его порядка. Представление чисел в естественной и показательной формах и примеры их отображения на экране индикатора приведены на рис. 1.4.

При показательной форме десятичное число N представляется в виде

$$N = \pm M \cdot 10^{\pm p},$$

где M — неправильная или правильная десятичная дробь (не более 8 цифр), называемой мантиссой числа N ; p — двузначное число, называемое порядком этого числа.

Знак мантиссы (являющийся и знаком числа N) занимает 1-е знакоместо слева, а мантисса — 2—9-е знакоместа. Знак порядка занимает 10-е знакоместо, а две цифры порядка — 11-е и 12-е знакоместа.

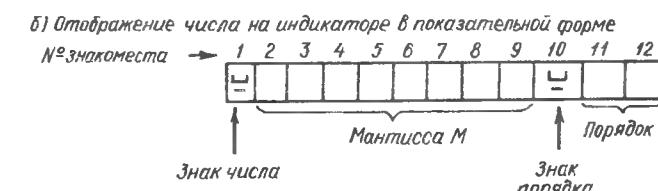
При вводе числа в показательной форме мантисса набирается точно так же, как и число в естественной форме представления, а именно, как любая неправильная десятичная дробь с положением десятичной точки в любой позиции. Точно так же она отображается на экране индикатора. Такое представление и отображение мантиссы называется ненормализованным. Мантисса результата выполнения какой-либо операции всегда отображается в нормализованном виде, т. е. десятичная точка фиксируется после первой цифры мантиссы. Примеры нормализованного и ненормализован-



Примеры:

№ знакоместа → 1 2 3 4 5 6 7 8 9 10 11 12

Числа в естественной форме											
—	2	5	3.	4	5	6	8	5	—	—	—
—	0.	5	4	7	8	9	4	1	—	—	—
—	9	7	8	—	—	—	—	—	—	—	—
—	3	4	5	8	6.	0	5	—	—	—	—



Примеры:

Число	Отображение на индикаторе с мантиссой M	
	ненормализованной	нормализованной
$\pm 3545.4901 \cdot 10^{12}$	$\pm 3545.4901 \pm 12$	$\pm 3.5434901 \pm 15$
$-623.05711 \cdot 10^{-7}$	$-623.05711 - 07$	$-6.2305711 - 05$
$-0.26 \cdot 10^3$	-0.26 ± 03	-2.6 ± 02

Рис. 1.4. Формы и примеры отображения чисел на экране индикатора

ного отображения мантиссы на экране индикатора приведены на рис. 1.4.

Из сказанного следует, что отображение чисел на индикаторе как в естественной, так и в показательной форме производится с точностью до восьми десятичных знаков. Диапазон представления чисел при этом составляет:

в естественной форме ± 99999999 ;
в показательной форме $\pm 9.999999 \cdot 10^{\pm 99}$.

В состав индикатора входит также дешифратор. Он преобразует код числа, хранящийся во входном (индикаторном) регистре процессора, в напряжение, подаваемое на элементы знакомест индикатора, что приводит к высвечиванию десятичного представления этого числа.

Программируемые микрокалькуляторы включают в себя следующие устройства памяти:

1. Постоянное запоминающее устройство (ПЗУ), предназначенное для хранения микропрограмм, с помощью которых осуществляется после нажатия клавиш выполнение соответствующих команд.

2. Оперативное запоминающее устройство (ОЗУ), предназначенное для хранения числовых данных и программ пользователя ПМК.

Кроме того, в МК-52 имеется полу постоянное запоминающее устройство (ППЗУ), предназначенное для длительного хранения программ и данных при выключенном ПМК. (В других ПМК записанные в память программы пользователя и данные при выключении стираются.)

Представление числовых данных и программ в памяти ПМК. Во всех устройствах памяти ПМК для хранения данных и программ используется двоичная система счисления (точнее, двоично-десятичная). Под системой счисления понимается способ представления чисел с помощью цифр. Основанием системы счисления называется количество различных цифр, используемых для представления любого числа. Система счисления, которой мы обычно пользуемся в повседневной жизни, называется десятичной. В ней для представления любого числа используется 10 различных цифр (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) и ее основание равно 10. В двоичной системе счисления (основание равно 2) имеются только две цифры: 0 и 1. Использование такой системы счисления существенно упрощает техническую реализацию устройств ПМК и одновременно повышает их надежность. В самом деле, для технической реализации одного разряда памяти достаточно обеспечить лишь два его устойчивых состояния, одно из которых будет соответствовать нулю, а другое единице, например, включено—выключено, намагнитено в одну сторону — намагнитено в другую сторону, закрыто и т. п. Поэтому любая информация, записываемая и хранящаяся в памяти ПМК, всегда представлена в виде последовательности нулей и единиц, которые могут быть восприняты как двоичные числа.

Двоичное представление числовых данных и программ в памяти ПМК является внутренним представлением. Пользователь ПМК, за исключением случаев, когда необходимо выполнение логических операций над числами, может ничего не знать о нем. Он почти всегда имеет дело с внешним представлением десятичных чисел в естественной или показательной форме, обозначениями клавиш в соответствии с символикой на пульте (при вводе программ) или шестнадцатеричными кодами команд (при чтении их на экране индикатора). Переход от внешнего представления данных и программ к внутреннему и обратно производится автоматически, без участия пользователя ПМК.

Числовые данные в десятичной системе счисления записываются и хранятся в памяти ПМК путем представления каждой цифры четырехзначным двоичным кодом (табл. 1.2).

Знак числа (порядка) представляется двоичными кодами: 1010 (минус) и 1111 (пробел, что эквивалентно плюсу). Десятичная точка, определяющая целую часть числа от дробной, записывается последовательностью двух двоичных кодов, первый из которых соответствует нулю, а второй — знаку минус: 00001010. Пусть, например, имеется число — 34.58. После ввода его в память ПМК оно будет представлено в следующем виде:

1010	0011	0100	00001010	0101	1000
—	3	4		5	8

Как уже отмечалось, на экране индикатора ПМК имеется 12 знакомест, каждое из которых предусматривает возможность индикации одной десятичной цифры и десятичной точки или знака минус (пробела). Двоичный код десятичной точки занимает два знакоместа, поэтому в общем случае для хранения числа в памяти ПМК требуется не 12, а 14 знакомест. Тогда код десятичного числа, хранимого в памяти ПМК и выдаваемого на экран индикатора, будет иметь длину $14 \times 4 = 56$ двоичных разрядов. С помощью четырехзначного двоичного кода можно получить $2^4 = 16$ различных комбинаций нулей и единиц, т. е. можно обозначить не 10, а 16 различных цифр, и, следовательно, использовать шестнадцатеричную систему счисления. Эта система применяется в ПМК для обозначения адреса и индикации на экране кодов команд программы, хранящихся в памяти. Последовательность шестнадцатеричных цифр принято обозначать 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Поскольку на экране индикатора латинские буквы A, B, C, D, E, F не воспроизводятся, они заменяются другими символами, отображаемыми на экране индикатора. Двоичные коды шестнадцатеричных цифр и их отображения на экране индикатора приведены в табл. 1.3. Здесь номер по порядку соответствует десятичному значению шестнадцатеричной цифры.

Наименьшим элементом памяти является двоичный разряд, в котором может храниться либо нуль, либо единица. Емкость памяти принято измерять в более крупных единицах, называемых байтами, каждый из которых включает в себя 8 двоичных разрядов. Таким образом, в одном байте могут быть записаны две

Таблица 1.2
Представление десятичных цифр двоичными кодами

Цифра	Двоичный код	Цифра	Двоичный код
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Таблица 1.3

Двоичные коды шестнадцатеричных цифр

Десятичные цифры	Шестнадцатеричные цифры	Отображение на экране индикатора	Двоичные коды
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	1111
8	8	8	1000
9	9	9	1001
10	A	—	1010
11	B	Б	1011
12	C	С	1100
13	D	Г	1101
14	E	Е	1110
15	F	—(пробел)	1111

шестнадцатеричные (и тем более две десятичные) цифры или одна десятичная точка.

Постоянное запоминающее устройство ПЗУ предназначено для хранения микропрограмм, с помощью которых осуществляется выполнение каждой команды. Оно допускает только считывание постоянно хранимых в нем микропрограмм, записываемых при изготовлении ПМК. Содержимое ПЗУ недоступно для пользователя ПМК. В ручном режиме при нажатии клавиши происходит обращение к микропрограмме, хранящейся в ПЗУ и выполняющей соответствующую команду ПМК. Выполнение микропрограммы начинается сразу после нажатия клавиши. Останов ПМК происходит сразу после завершения работы микропрограммы, реализующей соответствующую команду. При работе в автоматическом режиме останова ПМК после завершения выполнения микропрограммы не происходит, а осуществляется автоматический переход к выполнению следующей очередной команды программы с помощью соответствующей микропрограммы, хранящейся в ПЗУ. Процесс продолжается до тех пор, пока в программе пользователя не встретится команда останова С/П.

Оперативное запоминающее устройство (ОЗУ) предназначено для хранения программ и данных с обеспечением быстрого (оперативного) доступа к ним. В отличие от ПЗУ оно доступно для пользователя ПМК и служит оперативной памятью.

В ЭВМ всех типов данные и программы могут размещаться в любом месте оперативной памяти. В ПМК оперативная память жестко разделена на две половины по 105 байт в каждой (в МК-54

по 98 байт). В первой из них размещаются только числовые данные (ОЗУ данных, ОЗУД), а во второй — только программы пользователя (ОЗУ программ, ОЗУП).

ОЗУ данных (ОЗУД) предназначено только для хранения числовых данных. Для размещения одного числа максимальной длины, на которую рассчитан ПМК, требуется 14 знакомест или $4 \times 14 = 56$ двоичных разрядов, что составляет $56/8 = 7$ байт в ОЗУ данных. Таким образом в ОЗУ данных емкостью в 105 байт можно разместить одновременно $105/7 = 15$ различных чисел (в МК-54 $98/7 = 14$). В соответствии с этим ОЗУ данных подразделяется на 15 регистров (в МК-54 на 14) по 7 байт, в каждом из которых в любой момент времени может храниться одно и только одно число. Все регистры пронумерованы шестнадцатеричными числами: 0, 1, 2, ..., 9, А, В, С, Д, Е (в МК-54 регистр Е отсутствует). Номер регистра принято называть адресом, по которому производится обращение к регистру. Для того чтобы прочитать число, записанное в регистр памяти, или записать число, индицируемое на экране, достаточно кроме команды чтения (записи) указать одну шестнадцатеричную цифру — адрес соответствующего регистра. Поэтому ОЗУ данных в инструкции, прилагаемой заводом-изготовителем к каждому ПМК, называют адресуемой памятью, что не совсем точно, так как ОЗУ программ также является адресуемой памятью.

ОЗУ программ (ОЗУП) предназначено для хранения программ пользователя, выполняемых при решении задач в автоматическом режиме. В отличие от ОЗУ данных здесь адресуемой частью памяти является поле не в 7 байт, а в один. При этом в отличие от ОЗУД все 105 байт ОЗУП (в МК-54 98 байт) пронумерованы в десятичной системе счисления от 0 до 99, а затем .0, .3, .3, .4 (в МК-54 от 0 до 97).

Команды программы, предписывающие выполнение определенных законченных действий, могут быть закодированы либо двумя, либо четырьмя шестнадцатеричными цифрами. Каждый байт ОЗУП рассчитан на помещение в нем двух шестнадцатеричных цифр, являющихся кодом шага программы. Если код команды содержит две шестнадцатеричные цифры и размещается в одном байте ОЗУП, то такая команда называется одношаговой. Если же код команды содержит четыре шестнадцатеричные цифры и размещается в двух байтах ОЗУП, то такая команда будет двухшаговой. В этом случае адресом команды будет считаться адрес ее первого байта. При вводе программы с пульта ПМК код каждого очередного ее шага записывается в байт с очередным адресом. Программа, записанная в ОЗУП, выполняется в порядке возрастания адресов команд. Шестнадцатеричные коды команд, записываемые в программную память, приведены в табл. 1.3. Содержимое всех байтовых ячеек ОЗУП может быть просмотрено на экране индикатора, на котором одновременно отображаются коды трех одношаговых команд, предшествующих команде, адрес которой дополн-

нительно отображается в правой части экрана. Напомним, что, говоря о командах, мы подразумеваем длину каждой из них равной одному шагу, соответствующему нажатию одной непрефиксной клавиши. Длина программы, определяемая количеством команд, также подразумевается в одношаговом исчислении. Использование двухшаговых команд в каждом случае оговаривается особо.

Полупостоянное запоминающее устройство (ППЗУ). В большинстве ПМК при отключении электропитания автоматически стираются программы и данные, хранящиеся в оперативной памяти. При необходимости повторного решения задачи на ПМК после его выключения приходится вновь вводить программу в оперативную память вместе с исходными данными, что является большим неудобством. С целью его устранения в МК-52 введено дополнительно полупостоянное запоминающее устройство ППЗУ емкостью 512 байт, в котором после выключения калькулятора программы могут сохраняться до 5000 ч. Поэтому ППЗУ называют еще энергозависимой, или неразрушающей памятью.

Функционирует ППЗУ в трех режимах: стирания, записи и считывания. Операции в этих режимах всегда производятся над частью накопителя ППЗУ, называемого полем. Границы поля задаются адресом его начальной строки длиной в байт. Для указанного числа байт в командах стирания, записи и считывания отведено два знакоместа (две десятичных цифры), поэтому длина поля не должна превосходить 99 байт. Длина поля определяет число стираемых строк накопителя ППЗУ в режиме стирания, а также количество байт памяти с программами или данными, которыми обмениваются ОЗУ и ППЗУ в режимах записи и считывания. В последнем случае данные записываются в ОЗУ или считаются из него, начиная с нулевого регистра ОЗУ данных, а программы, — начиная с нулевого байта ОЗУ программ, имеющих адрес 00.

Существенным недостатком ППЗУ является значительное потребление энергии в названных режимах от автономного источника питания, что вынуждает обращаться к ППЗУ практически только при его питании от сети. Поэтому в большинстве выпускаемых ПМК ППЗУ отсутствует.

1.2.3. ПРОЦЕССОР (ВЫЧИСЛИТЕЛЬ)

Процессор содержит арифметико-логическое устройство, непосредственно выполняющее операции над числами, и операционное устройство, предназначенное для хранения чисел и результатов вычислений. Кроме того, процессор содержит счетчик адреса команд (СЧАК), точнее, счетчик адреса шага программы, функционирующий при вводе и выполнении программы, записанной в программной памяти ПМК. После выполнения или ввода очередной команды в СЧАК добавляется единица и, таким образом,

содержимое СЧАК указывает номер (адрес) очередной команды, которая должна быть введена или выполнена.

Операционное устройство состоит из четырех регистров: X, Y, Z, T.

Регистр X, называемый входным или индикаторным, предназначен для запоминания и хранения числа, набранного с пульта управления, или числа, являющегося результатом выполнения какой-либо операции. Данный регистр соединен через дешифратор с индикатором. Таким образом, содержимое регистра X всегда отображается на экране индикатора.

Регистр Y служит для помещения в него второго числа перед выполнением двухместной операции. После ее выполнения результат операции заносится в регистр X. Старое содержимое регистров X и Y при этом стирается. Эти регистры называются операционными, так как в них хранятся числа, над которым производятся операции.

Регистры Z и T вместе с регистрами X и Y образуют стековую, или магазинную память, работа которой будет рассмотрена подробно в главе, посвященной ее практическому использованию.

Кроме регистров операционного устройства в процессоре имеется вспомогательный регистр X1 (регистр предыдущего результата), который служит для запоминания и хранения результата предыдущей операции. Он необходим, поскольку значения операндов в регистрах X и Y стираются после выполнения операции.

Числа в регистрах процессора представляются с помощью двоичных кодов десятичных цифр. В случае выполнения операций сложения и вычитания над числами в показательной форме предварительно производится выравнивание порядков.

Устройство управления (УУ) служит для согласования во времени работы всех элементов ПМК и конструктивно совмещено с процессором.

1.2.4. ВНЕШНИЕ УСТРОЙСТВА

В МК-52 предусмотрен интерфейс для подключения внешних устройств. Под интерфейсом понимается совокупность унифицированных шин (проводов), сигналов и электронных схем их преобразования из сигналов устройств ПМК. Выходом интерфейса для подключения внешних устройств служит разъем, состоящий из 22 гнезд, расположенный справа и закрытый пластмассовой крышкой (другой разъем из 16 гнезд, расположенный слева от этого разъема, — технологический). К разъему могут быть подключены самые различные внешние устройства, разработанные под интерфейс. К числу таких устройств относятся блоки расширения памяти БРП-2 и БРП-3, представляющие собой дополнительную па-

мять для хранения прикладных программ. Эти блоки приобретаются за отдельную плату. Описание и инструкция по эксплуатации прилагаются к каждому из них.

1.3. ФУНКЦИОНИРОВАНИЕ ПМК ПРИ ВЫПОЛНЕНИИ ТИПОВОЙ КОМАНДЫ

Ввод числа в ПМК. Ввести число в ПМК — значит занести его в регистр X процессора, содержимое которого всегда индируется на экране. Ввод числа осуществляется путем набора его цифр и знаков (минус и десятичная точка) на клавиатуре пульта управления. Цифры набираются последовательно, начиная со старшего разряда. Целая часть числа отделяется от дробной путем ввода в соответствующем месте точки (нажатием клавиши .). В случае ввода отрицательного числа минус вводится после набора последней цифры самого младшего разряда путем нажатия клавиши / —/ (команда изменить знак числа).

Пусть, например, нам необходимо ввести в ПМК отрицательное число, равное —253. В процессе его ввода устройства ПМК будут функционировать следующим образом:

1. С пульта управления вводится первая цифра числа (2).
2. Устройство управления запускает микропрограмму ввода, хранящуюся в ПЗУ.

3. В соответствии с микропрограммой ввода первая цифра записывается в старший разряд регистра X процессора.

4. Код числа, записанного в регистре X с помощью дешифриатора, преобразуется в напряжение, соответствующее первой цифре числа. Это напряжение будет подано на второе знакоместо индикатора (первое знакоместо отведено для хранения знака числа). Эта цифра отобразится на экране индикатора (в данном случае цифра 2).

5. С пульта вводится вторая цифра числа (5). Процесс повторяется с той разницей, что микропрограмма ввода поместит ее в следующий разряд регистра X и соответственно отобразит на 3-м знакоместе индикатора. Третья цифра (3) будет соответственно занесена в 4-й разряд регистра X и отобразится на 4-м знакоместе индикатора. Таким образом, на экране индикатора будут последовательно отображаться цифры:

- 2.
- 25.
- 253.

6. Для положительного числа работа микропрограммы ввода заканчивается после ввода последней цифры числа (самого младшего его разряда). Если число отрицательное, то знак минус вводится после набора всех цифр числа. Нажимается клавиша /—/ и происходит обращение к микропрограмме «Изменить знак», которая заносит в знаковый разряд регистра X минус (если до этого

там находился пробел) или пробел (если до этого там находился знак минус). Соответственно происходит индикация на первом знакоместе индикатора.

Итак, в случае набора на пульте управления ПМК числа —253 на экране индикатора будут последовательно отображаться:

- 2.
- 25.
- 253.
- 253.

Выполнение одноместных операций. Для выполнения одноместных операций (вычисления функции одного аргумента x) значение операнда (аргумента x) должно быть предварительно помещено в регистр X. По команде, предписывающей выполнение одноместной операции (например, после нажатия клавиши sin), выполняются следующие действия:

1. Устройство управления обращается к микропрограмме выполнения одноместной операции, постоянно хранящейся в ПЗУ (микропрограмме вычисления sin).

2. Используя содержимое регистра X в качестве исходного данного, процессор в соответствии с микропрограммой вычисляет значение функции для аргумента x ($\sin x$). При этом первоначальное содержимое регистра X (аргумент x) засыпается в регистр предыдущего результата X1. Результат вычисления функции ($\sin x$) помещается в регистр X процессора.

3. Содержимое регистра X — значение функции аргумента x ($\sin x$) отображается на экране индикатора.

Выполнение двухместной операции. Для выполнения двухместной операции необходимо ввести в операционное устройство процессора два числа: одно — в регистр X, другое — в регистр Y. Ввод двух чисел с клавиатуры пульта управления осуществляется путем последовательного их набора. В качестве разделения чисел используется команда B↑. По этой команде введенное и индицируемое на экране первое число будет помещено в регистр Y, а в регистр X после этого можно будет ввести второе число.

Пусть, например, требуется вычесть из числа A число B. Тогда необходимо на пульте управления ПМК сначала набрать число A, нажать клавишу B↑, а затем набрать число B. В результате этого число A будет занесено в регистр Y, а число B — в регистр X. Если числа для выполнения двухместной операции вводятся из оперативной памяти ПМК, никаких дополнительных команд, осуществляющих разделение чисел, не требуется (они уже разделены тем, что хранятся в различных регистрах памяти). При этом первое число (A) вводится в регистр X, а при вводе второго числа первое выталкивается в регистр Y, а на его месте регистр X записывает второе число (B).

По команде, предписывающей выполнение двухместной операции (например, вычитание после нажатия клавиши —) происходят следующие действия:

1. Устройство управления обращается к микропрограмме, хранящейся в ПЗУ и выполняющей соответствующую операцию (вычитание).

2. Процессор выполняет микропрограмму, реализующую заданную операцию (вычитание). При этом:

содержимое регистра X (вычитаемое) помещается в регистр результата предыдущего действия (X_1);

из содержимого регистра Y вычитается содержимое регистра X, результат (разность) помещается в регистр X, а старое его содержимое стирается.

3. Результат операции (разность) индицируется на экране.

Выполнение команды в автоматическом режиме. Пусть нам необходимо выполнить операцию вычитания. Полагаем, что в регистр Y помещено уменьшаемое, а в регистр X — вычитаемое. Предположим, что в счетчике адреса команды (СЧАК) находится число — адрес очередной i -й команды, которую необходимо выполнить. Процесс выполнения i -й команды будет протекать следующим образом:

1. Управляющее устройство обращается к СЧАК.

2. По содержимому СЧАК определяется адрес i -й очередной команды, которую следует выполнить.

3. По i -му адресу производится обращение к команде программы, хранящейся в ОЗУП.

4. В соответствии с командой программы производится обращение к микропрограмме, хранящейся в ПЗУ, которая реализует выполнение i -й команды.

5. Содержимое регистра X (вычитаемое) засыпается в регистр предыдущего действия X_1 .

6. Из содержимого регистра Y вычитается содержимое регистра X. Результат (разность) помещается в регистр X.

7. Значение разности отображается на экране индикатора до момента занесения в регистр X нового содержимого при выполнении $(i + 1)$ -й команды.

8. В счетчик адреса команд добавляется единица, его содержимое после этого будет соответствовать адресу $(i + 1)$ -й команды. С этого момента ПМК подготовлен к выполнению следующей команды. Переход к ее выполнению осуществляется автоматически. Далее снова выполняется пункт 1.

9. Процесс повторяется до тех пор, пока не встретится команда останова С/П, после чего (т. е. после останова вычислений по программе) в счетчике адреса команд будет адрес следующей, уже несуществующей команды. Таким образом, по окончании работы программы в СЧАК будет находиться число, на единицу большее числа команд программы. Для возврата СЧАК в исходное (нулевое) состояние следует нажать клавишу В/О (возврат обратно).

Глава 2. ОСНОВНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ОПЕРАЦИИ, ВЫПОЛНЯЕМЫЕ ПМК

Освоение ПМК начинается с умения включить его, произвести ввод чисел и выполнить над ними основные вычислительные операции.

Это, во-первых, одноместные операции, к которым относятся операции вычисления значений функций по введенному в ПМК значению аргумента x : $\sin x$, $\cos x$, $\tg x$, $\arctg x$, $\ln x$, e^x . При этом вычисляется восьмизначное значение функции при гарантированной точности в шесть знаков.

Научившись выполнять одноместные операции, мы как бы получаем в свое распоряжение очень удобную малогабаритную таблицу часто встречающихся функций, к которой можно обращаться без всяких интерполяций и учета поправок. Для вычисления функции достаточно ввести в ПМК значение аргумента, нажать клавишу с наименованием функции и прочитать ее значение на экране индикатора.

Об эффективности применения ПМК вместо печатных таблиц функций говорит тот факт, что известные семизначные таблицы десятичных логарифмов Вега занимают том в 500 страниц. Тогда для всех таблиц функций, содержащих все вычисляемые на ПМК значения, потребовалось бы не менее 10 таких томов.

Во-вторых, это двухместные операции, к которым относятся арифметические операции над двумя числами: сложение, вычитание, умножение, деление, возведение в степень с выдачей результата на экран индикатора. Такие операции очень часто приходится выполнять на практике. Для этого достаточно ввести в ПМК два числа, нажать клавишу с названием арифметической операции и прочитать на экране индикатора искомый результат.

В данной главе будет показано, как практически выполняются с помощью ПМК основные вычислительные операции.

2.1. ВКЛЮЧЕНИЕ ПМК, ВВОД И ИНДИКАЦИЯ ЧИСЛА

Включение ПМК. Для включения ПМК необходимо установить переключатель, расположенный слева под экраном индикатора, в правое положение. В этом положении на переключателе будет видна красная точка. Тогда на экране индикатора в старшем цифровом разряде появится нуль, что свидетельствует о готовности ПМК к работе. Если же при этом на месте всех знаков индикатора будут высвечиваться точки, то это будет означать, что источник питания ПМК разряжен и требуется его зарядка или замена.

Если ПМК был ранее включен и использовался для других вычислений, то показания индикатора необходимо сбросить путем нажатия клавиши Сх. Повторное включение ПМК допускается не раньше чем через 10 с после его выключения, иначе не успеет произойти обнуление памяти МК. В случае кратковременного

прерывания питания ПМК (например, если плохой контакт выпрямителя при питании от сети) необходимо его выключить и снова включить через 10—15 с.

Ввод и отображение чисел на экране индикатора. ПМК оперирует с положительными и отрицательными числами в показательной форме. При этом ввод и отображение на индикаторе чисел в диапазоне от 10^{-99} до 10^8 — 1 (чисел, содержащих не более 8 цифр) осуществляются в естественной форме — в виде правильной или неправильной десятичной дроби. Если же число содержит более 8 знаков, то оно вводится и отображается в показательной форме.

Например, восьмизначное число -494751.23 будет отображаться на индикаторе следующим образом:

—494751.23
↑ ↑
целая дробная
часть часть
знак десятичная
числа точка

В случае положительного числа знак минус перед старшим разрядом его будет опущен.

Если же число состоит из более чем 8 цифр, например, -1234567890 (10 цифр), то такое число в естественной форме не может быть введено и отображено на экране индикатора. При попытке ввести это число будет введено и отображено на экране индикатора восьмизначное число -12345678 . На ввод последних цифр (90) ПМК не будет реагировать. Такое число необходимо представить в виде восьмизначной мантиссы и порядка — показателя степени десятичного основания: $-1234567890 = -1234568 \times 10^3$. При вводе такого числа сначала вводится мантисса, затем — порядок и на экране индикатора будет отображено

—1234568 03
↑ ↑
манти́сса порядок
знак
числа

После выполнения ПМК какой-либо операции мантисса числа нормализуется, т. е. положение десятичной точки в ней устанавливается после старшего разряда мантиссы. Соответственно изменяется и порядок. Например, приведенное выше число будет отображено так:

—1.234568 09
нормали-
зованный порядок
манти́сса

Числа, меньшие единицы, после проведения какой-либо операции также будут отображаться с нормализованной мантиссой во всех случаях, независимо от того, сколько знаков было в исходном,

введенном числе. Так, число -0.0123456 при вводе будет отображаться на индикаторе в том виде, в котором оно было введено, а после выполнения какой-либо операции станет отображаться так:

—1.23456 —02
↑ ↑
манти́сса порядок
знак
числа знак
порядка

Знак числа и знак порядка вводится и отображается на экране только для отрицательных мантисс и порядков (у положительных они опускаются). Для значения порядка отводятся две десятичные цифры, следовательно, порядок числа может принимать значения от -99 до $+99$.

Для ввода чисел в ПМК используются клавиши: серого цвета:

0, 1, 2, ..., 9 — цифровые клавиши,
. — десятичная точка;
красного цвета:

Сх — сброс регистра Х (показаний индикатора), предписывающие выполнение соответствующих команд.

Поэтому порядковый номер нажимаемой клавиши будет соответствовать порядковому номеру выполняемой команды. Порядковые номера команд здесь и в дальнейшем условимся обозначать двузначными десятичными числами, начиная с 00. Содержимое регистра Х, которое всегда отображается на экране индикатора ПМК, будем обозначать при этом в круглых скобках после обозначения команды: (...).

Пусть, например, требуется ввести число -148.12 . Ввод его осуществляется путем последовательного набора следующих клавиш:

Порядковый номер нажимаемой клавиши	Нажимаемая клавиша	Индикация на экране после нажатия клавиши
00.	1	1.
01.	4	14.
02.	8	148.
03.	.	148.
04.	i	148.1
05.	2	148.12
06.	/—/	—148.12

Знак отрицательного числа / —/ вводится после набора всех его цифр, а знак отрицательного порядка (с помощью той же клавиши /—/) — после набора всех цифр порядка. Если при вводе числа будет сделана ошибка, то следует нажать клавишу Сх (клавишу обнуления регистра Х), после чего на экране индикатора появится нуль, и тогда можно будет заново набрать введенное число.

Для ввода числа с порядком необходимо:

ввести мантиссу;

нажать клавишу ВП (ввод порядка);

ввести порядок;

в случае отрицательного порядка нажать на клавишу $/-$.

Пусть необходимо ввести число $-148.12 \cdot 10^{-15}$. Для этого необходимо ввести последовательность команд:

00.1 (1)	04. 1 (148.1)	08. 1 (-148.12.01)	5) 3.628000193	9) $0.125 \cdot 10^{-12}$
01. 4 (14)	05. 2 (148.12)	09. 5 (-148.12_15)	6) 1234567890	10) $421.65 \cdot 10^{97}$
02. 8 (148)	06. /—/ (-148.12)	10. /—/ (-148.12—15)	7) 0.123456789	11) $421.65 \cdot 10^{98}$
03. . (148.)	07. ВП (-148.12_00)		8) 448.35	12) $0.245 \cdot 10^{-99}$

Запись последовательности нажимаемых клавиш при вводе числа слишком громоздка и малообозрима. Поскольку ввод любого числа с пульта ПМК производится по строго определенным правилам, изложенным выше, то команды ввода, выполняемые вручную, можно не перечислять, а заменить все их одним предписанием ВВЕСТИ $N(x, y)$, где N — любое число или наименование переменной, а x и y — содержимое регистров X и Y после ввода числа N (указывать x, y не обязательно). С помощью предписания ВЫДАТЬ будем аналогично записывать операцию чтения результата пользователем с экрана индикатора и его фиксацию на бумаге. Предписания ВВЕСТИ и ВЫДАТЬ (можно использовать сокращенные обозначения ВВ и ВЫ) выполняются пользователем ПМК только вручную. Такие предписания иногда называют директивами. Эти предписания не входят в число команд и не нумеруются, поскольку их выполнение не входит непосредственно в процесс вычислений. Указанные предписания записываются в тексте наряду с командами с отступом на 2—3 позиции вправо от номеров команд. Тогда запись ввода числа $-148.12 \cdot 10^{-15}$ и чтение его на экране индикатора будет выглядеть следующим образом:

ВВЕСТИ — 148.12 · 10⁻¹⁵ (-148.12 —15)

ВЫДАТЬ — 148.12 · 10⁻¹⁵

Некорректные операции ввода. Максимальное число, которое может быть введено в регистр X с клавиатуры ПМК, равно $9.999999 \cdot 10^{99}$. Если будет сделана попытка ввести число, большее указанного значения, например $99.99999 \cdot 10^{99}$, то после набора порядка на экране индикатора вместо мантиссы числа появится сигнал ошибки ЕГГОГ. В этом случае необходимо сбросить содержимое регистра X (нажав клавишу Сх) и ввести число, по своему значению не превышающее максимального допустимого.

Упражнения

2.1.1. Что будет отображено на экране индикатора при попытке непосредственно ввести следующие числа?

- | | | |
|----------|----------------|----------------------------|
| 1) 83 | 5) 3.628000193 | 9) $0.125 \cdot 10^{-12}$ |
| 2) —0.01 | 6) 1234567890 | 10) $421.65 \cdot 10^{97}$ |
| 3) 0.01— | 7) 0.123456789 | 11) $421.65 \cdot 10^{98}$ |
| 4) 0.1— | 8) 448.35 | 12) $0.245 \cdot 10^{-99}$ |

2.1.2. Как необходимо преобразовать следующие числа, чтобы их приближенные значения можно было ввести в память ПМК?

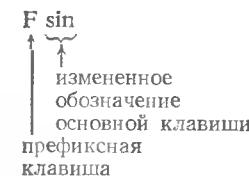
- | | |
|----------------|----------------------------|
| 1) 3.628000193 | 5) 0.000000012 |
| 2) 1234567890 | 6) $421.65 \cdot 10^{98}$ |
| 3) 0.123456789 | 7) $0.245 \cdot 10^{-100}$ |
| 4) 0.000000091 | |

2.2. ОДНОМЕСТНЫЕ ОПЕРАЦИИ

Одноместные операции выполняются на ПМК только над числами, предварительно помещенными в регистр X и отображаемыми на экране индикатора. Результат операции (значение вычисленной функции) помещается в тот же регистр X на место аргумента и отображается на экране индикатора. Старое содержание регистра X (аргумент x) при этом стирается, но его можно восстановить, поскольку перед вычислением функции значение аргумента x помещается в регистр предыдущего результата X_1 .

Клавиши, используемые для выполнения одноместных операций. При выполнении одноместных операций в качестве префиксных используются клавиши F (желтого цвета) или клавиши K (голубого цвета, расположенные под клавишей F). После нажатия клавиши F или K основное их название и назначение будет изменено. Обозначение клавиши следует читать тогда не на самой клавише, а над ней: желтого цвета, если перед этим была нажата клавиша F, или голубого, если была нажата клавиша K.

Последовательное нажатие сначала префиксной клавиши, а затем основной и выполнение после этого соответствующей операции условимся рассматривать как одну команду и записывать слитно:



Имеются также две одноместные операции, которые предписываются к выполнению без использования префиксных клавиш. К ним относятся изменение знака числа (клавиша $/—/$) и сброс неправильно введенного в регистр X числа (красная клавиша Сх). Перечень одноместных операций и клавиш, предназначенных для их реализации, приведен в табл. 2.1.

Одноместные операции ПМК

Наименование операции	Обозна- чение клавиши	Отображение на экране индикатора	
		До нажатия клавиши	После нажатия клавиши
Вычисление тригонометрических и других функций	F sin F cos F tg F \sin^{-1} F \cos^{-1} F \tg^{-1} F ex F ln F 10^x F lg F \sqrt{x} F x^2 F $1/x$ F π	x x x x x x x x x x x x x x 3,1415926	sin x cos x tg x $\arcsin x$ $\arccos x$ $\operatorname{arctg} x$ e^x $\ln x$ 10^x $\lg x$ \sqrt{x} x^2 $1/x$ Безразлично
Восстановление результата предыдущего действия	FBx	Результат любой операции	Содержимое регистра X, которое было до выполнения операции
Сброс ошибочно нажатой префиксной клавиши	FCF	Безразлично	Безразлично
Выделение целой части числа	K [x]	999.99999	999
Выделение дробной части числа	K {x}	999.99999	0.99999
Определение абсолютного значения числа	K x	99999999 —99999999	99999999 99999999
Определение знака числа	KZN	99999999 0 —99999999	1 0 —1
Перевод градусов и минут угла в градусы и доли градуса	K° или K, \rightarrow о	9 9 9.5 9 9 9 9 град. мин доли мин	9 9 9 9 9 9 9 9 град. доли град.
Перевод градусов и долей градуса угла в градусы и минуты	K° или K, \leftarrow о	9 9 9.9 9 9 9 9 град. доли град.	9 9 9.5 9 9 9 9 град. мин доли мин
Перевод часов, минут и секунд в часы и доли часа	K°° или K,, \leftarrow о	9 9 9.5 9 5 9 9 ч мин с	9 9 9.9 9 9 9 9 ч доли ч

Наименование операции	Обозна- чение клавиши	Отображение на экране индикатора	
		До нажатия клавиши	После нажатия клавиши
Перевод часов и долей часа в часы, минуты и секунды	K°° или K,, \leftarrow о	9 9 9.9 9 9 9 9 ч доли ч	9 9 9.5 9 5 9 9 ч мин с
Генерация случайного числа	KСЧ	Безразлично	Случайное число в интервале (0, 1)
Логическое отрицание	КИНВ	8.9 9 9 9 9 9 9 ↑ аргумент	8.F F F F F F F ↑ результат
Результат — шестнадцатиричное число, полученное инверсией нулей и единиц двоичного представления десятичного аргумента	признак логич. операции	признак логич. операции	признак логич. операции
Сброс неправильно введенного в регистр X числа	Cx	Безразлично	0
Изменение знака числа	/—/	999999999 —999999999	—999999999 999999999

П р и м е ч а н и я: 1.9 — признак любой десятичной цифры от 0 до 9; 5 — признак любой десятичной цифры от 0 до 5; F — признак любой шестнадцатиричной цифры от 0 до F (см. табл. 1.3).
2. Все операции с префиксной клавишей K только для МК-61 и МК-52.

2.2.1. ПРИМЕРЫ ВЫПОЛНЕНИЯ ОДНОМЕСТНЫХ ОПЕРАЦИЙ

Вычисление тригонометрических, показательных, логарифмических и алгебраических функций. Для вычисления тригонометрических, показательных, логарифмических и алгебраических функций заданного аргумента x необходимо:

в случае вычисления тригонометрических функций установить переключатель Р—ГРД—Г (радианы—грады—градусы) в положение, соответствующее выбранной единице измерения аргумента x ;

набрать на пульте значение аргумента и получить его отображение на экране индикатора;

нажать префиксную клавишу F;

нажать клавишу с обозначением вычисляемой функции;

прочитать результат на экране индикатора.

Покажем на конкретных примерах, какие клавиши и в какой последовательности следует нажимать, чтобы получить искомый результат. Числовые значения, отображаемые на экране индика-

тора после нажатия каждой клавиши, будем помещать в круглых скобках правее обозначения каждой команды.

Примеры:

Вычислить значения следующих функций:

- | | |
|---------------------------------------|---|
| 1) $\sin 32^\circ$ | 4) $\lg 412$ |
| ВВЕСТИ 32 (32) | ВВЕСТИ 412 (412) |
| 00. F sin (5.2991926 —01) | 00. F lg (2.6148971) |
| ВЫДАТЬ 5.2991926. 10 ⁻¹ | ВЫДАТЬ 2.6148971 |
| 2) $10^{-1.48}$ | 5) $\arcsin 0.975$ |
| ВВЕСТИ —1.48 (-1.48) | ВВЕСТИ 0.975 (0.975) |
| 00. F 10 ^x (3.3113114 —02) | 00. F sin ⁻¹ (1.346721 радианы или 77.161431°) |
| ВЫДАТЬ 3.3113114 —02 | если Р—ГРД—Г в положении Г
то ВЫДАТЬ 77.161431° |
| 3) $e^{-0.48}$ | если Р—ГРД—Г в положении Р
то ВЫДАТЬ 1.346721 |
| ВВЕСТИ —0.48 (-0.48 —01) | |
| 00. Fe ^x (6.1878339 —01) | |
| ВЫДАТЬ 6.1878339. 10 ⁻¹ | |

Вычисление значений нескольких функций одного аргумента. Использование результата предыдущего действия. Часто требуется для одного значения аргумента определить значения нескольких функций, например $\sin x$, $\cos x$ и др. Трудность заключается в том, что результат вычисления функции помещается в тот же регистр X, что результат вычисления аргумента, которое после куда было ранее помещено значение аргумента, которое после выполнения одноместной операции стирается в регистре X. Для восстановления значения аргумента в регистре X после вычисления функции достаточно нажать клавишу FBx, после чего будет выполнена команда восстановления результата предыдущего действия. При этом значение аргумента будет считано из регистра X1. При этом значение аргумента Х1 в регистре X, а значение результата предыдущего действия X1 в регистре Y.

Пример:

Вычислить $\sin 32^\circ$, $\cos 32^\circ$, $\tg 32^\circ$.
Р—ГРД—Г в положении Г (градусы).

- | |
|--------------------------------|
| ВВЕСТИ 32 (32) |
| 00. Fsin (5.2991926 —01) |
| ВЫДАТЬ sin 32° = 5.2991926 —01 |
| 01. FBx (32, 5.2991926 —01) |
| 02. Fcos (8.4804814 —01) |
| ВЫДАТЬ cos 32° = 8.4804814 —01 |
| 03. FBx (32, 8.4804814) |
| 04. Ftg (6.2486932 —01) |
| ВЫДАТЬ tg 32° = 6.2486932 —01 |

Прямой и обратный перевод значений угла «градусы, минуты \rightarrow градусы, доли градуса» (для МК-61 и МК-52). Значение угла в градусах и минутах должно быть отображено на экране индикатора в виде десятичной дроби, целая часть которой определяет число градусов, две цифры правее точки — число минут, а остальные цифры справа от минут — доли минут. Максимальное число цифр в представлении угла равно 8. Если под символом 9

подразумевать любую десятичную цифру от 0 до 9, а под символом 5 — от 0 до 5, то отображение угла в градусах и минутах будет представлено в виде

999.59999
↑ ↑ ↑
| | |
дели минуты
минуты
градусы

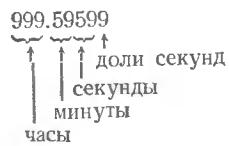
Например, запись 23.314568 означает $23^\circ 31.4568$. Значение угла в градусах и долях градуса представляется в обычной десятичной дроби, число десятичных цифр в которой также не должно превосходить 8.

Перевод значений «градусы, минуты \rightarrow доли градуса» осуществляется с помощью команды K^{\rightarrow} , а обратный перевод «градусы, доли градуса \rightarrow градусы, минуты» — с помощью команды K^{\leftarrow} . Следует отметить, что обозначения клавиш K^{\rightarrow} и K^{\leftarrow} выбраны не совсем удачно. Из этих обозначений трудно понять, что во что переводится. Поэтому клавишу перевода «градусы, минуты \rightarrow градусы, доли градуса» будем обозначать $K, \rightarrow 0$, а клавишу обратного перевода «градусы, доли градуса \rightarrow градусы, минуты» так: $K, \leftarrow 0$. Помимо удобства печати в одну строку предлагаемые обозначения ясно показывают направлением стрелки, в какую сторону должно идти преобразование: от минут (запятая) к градусам или наоборот. В то же время предлагаемые обозначения не мешают чтению обозначений над клавишами, поскольку в них направления стрелок полностью совпадают. При переводе малых значений углов, выраженных в долях градуса, в минуты результат операции отображается на индикаторе в показательной форме. В этом случае его следует записать в естественной форме. Тогда первые две цифры правее точки будут определять число минут. При выполнении операций перевода «градусы, минуты — градусы, доли градуса» и обратно положение переключателя Р—ГРД—Г (радианы—грады—градусы) может быть любым. Однако если результат этих операций предполагается использовать в дальнейших вычислениях, то указанный переключатель следует поставить в положение Г (градусы).

Примеры:

- | | |
|---|---|
| 1. Перевести значения угла в градусах и минутах в градусы и доли градуса. | |
| 1) $120^\circ 15.6' =$ | 2) $0^\circ 00.6' =$ |
| ВВЕСТИ 120.156 (120.156) | ВВЕСТИ 0.006 (0.006) |
| 00. K, $\rightarrow 0$ (120.26) | 00. K, $\rightarrow 0$ (1 —02 = 0.01) |
| ВЫДАТЬ 120.26° | ВЫДАТЬ 0.01° |
| 2. Перевести значение угла в градусах и долях градуса в градусы и минуты. | |
| 1) $120.26^\circ =$ | 2) $0.01^\circ =$ |
| ВВЕСТИ 120.26 (120.26) | ВВЕСТИ 0.01 (0.01) |
| 00. K, $\leftarrow 0$ (120.156) | 00. K, $\leftarrow 0$ (0.6 —02 = 0.006) |
| ВЫДАТЬ 120° 15.6' | ВЫДАТЬ 0° 00.6' |

Прямой и обратный перевод значений времени «часы, минуты, секунды \leftrightarrow часы, доли часа» (для МК-61 и МК-52). Значения времени в часах, минутах и секундах для выполнения операций перевода должны быть представлены в виде десятичной дроби, где целая часть определяет количество часов, две цифры правее десятичной точки — число минут, две следующие цифры правее минут — число секунд, а последние цифры справа — доли секунд. Отображение значений времени на экране индикатора будет выглядеть следующим образом:



где 9 — любая цифра от 0 до 9; 5 — любая цифра от 0 до 5.

Например, 12.374415 означает 12 ч 37 мин 44.15 с. Количество цифр в значении времени не должно превышать 8. Значение времени в часах и долях часа представляется десятичной дробью, где целая часть обозначает число часов, а дробная — доли часа. При этом общее количество цифр в дроби также не должно превышать 8. Аргумент времени может быть как положительным, так и отрицательным. При малых значениях времени (доли часа) результат перевода отображается в показательной форме. Этот результат следует вручную перевести в естественную форму, после чего прочитать значения часов, минут и секунд.

Перевод значений «часы, минуты, секунды \rightarrow часы, доли часа» осуществляется с помощью команды K_o''' , а обратный перевод «часы, доли часа \rightarrow часы, минуты, секунды» — с помощью команды K_o''' . По аналогии с операциями перевода угла будем использовать в дальнейшем упрощенные обозначения указанных команд: $K_{...} \rightarrow 0$ (часы, минуты, секунды \rightarrow часы, доли часа) и $K_{...} \leftarrow 0$ (часы, доли часа \rightarrow часы, минуты, секунды).

Примеры:

1. Перевести заданное время в часах, минутах и секундах в часы и доли часа.

- | | |
|---|--|
| 1) 12 ч 50 мин 32,6 с = | 2) 0 ч 02 мин 15 с = |
| ВВЕСТИ 12.50326 (12.50326) | ВВЕСТИ 0.0215 (0.0215) |
| 00. К... \rightarrow 0 (12.842386) | 00. К... \rightarrow 0 (3.75 —02=0.0375) |
| ВЫДАТЬ 12.842386 ч | ВЫДАТЬ 0.0375 ч |
| 2. Перевести заданное время в часах в часы, минуты и секунды. | |
| 1) 12.8423861 ч = | 2) 0.0375 ч |
| ВВЕСТИ 12.8423861 (12.8423861) | ВВЕСТИ 0.0375 (0.0375) |
| 00. К... \leftarrow 0 (12.503259) | 00. К... \leftarrow 0 (2.15 —02=0.0215) |
| ВЫДАТЬ 12 ч 50 мин 32,59 с | ВЫДАТЬ 0 ч 02 мин 15 с |

Некорректные операции. В тех случаях, когда ПМК будет предписано выполнять некорректную операцию, на экране индикатора появится сигнал ошибки ЕГГОГ. К числу некорректных одноместных операций относятся попытки вычислений:

1) функций, аргументы которых находятся вне пределов допустимых значений, указанных в табл. 2.2;

Таблица 2.2

Относительная погрешность вычисления функций
и пределы допустимых значений аргумента

Наименование функции y	Максимальная относительная погрешность	Пределы допустимых значений аргумента x
$\sin x$	$3 \cdot 10^{-7}$	$1 \cdot 10^{-49} < x < 10^{10}$
$\cos x$	$3 \cdot 10^{-7}$	$1 \cdot 10^{-49} < x < 10^{10}$
$\operatorname{tg} x$	$3 \cdot 10^{-7}$	$\pi/2 + n\pi \neq 1 \cdot 10^{-49} < x < 10^{10}$
$\arcsin x$	$3 \cdot 10^{-7}$	$ x \leqslant 1$
$\arccos x$	$3 \cdot 10^{-7}$	$ x \leqslant 1$
$\operatorname{arctg} x$	$3 \cdot 10^{-7}$	$ x \leqslant 9.999999 \cdot 10^{99}$
e^x	$4 \cdot 10^{-7}$	$ x \leqslant 100 \ln 10 = 230.2581$
x^2	$1 \cdot 10^{-7}$	$ x < 10^{50}$
10^x	$4 \cdot 10^{-7}$	$ x \leqslant 99.99999$
\sqrt{x}	$1 \cdot 10^{-4}$	$x \geqslant 0$
$1/x$	$1 \cdot 10^{-7}$	$x \neq 0$
$\ln x$	$3 \cdot 10^{-7}$	$x > 0$
$\lg x$	$3 \cdot 10^{-7}$	$x > 0$

2) углов в градусах и долях градуса, а также времени в часах и долях часа, когда число минут или секунд в аргументе превышает 60;

3) когда результат операции превосходит $9.9999999 \cdot 10^{99}$.

После появления на экране сигнала некорректной операции ЕГГОГ можно осуществить ввод числа в регистр X (сигнал ЕГГОГ при этом стирается) и выполнять дальнейшие вычисления.

Затраты времени на выполнение одноместных операций. Каждый раз при нажатии клавиш, определяющих соответствующую одноместную операцию, производится вычисление функции от аргумента, который хранится в регистре X и отображается на экране индикатора. При этом из постоянного запоминающего устройства (ПЗУ) ПМК вызывается программа вычисления соответствующей функции на экране индикатора (в регистре X). Время вычисления функции, начиная от нажатия клавиши до получения результата на экране, зависит от характера функции и составляет в среднем:

для арифметических операций $1/x$, \sqrt{x} , x^2 — не более 0,5 с;
для вычисления функций $\ln x$, $\lg x$, e^x , $\sin x$, $\cos x$, $\tg x$,
 $\arccos x$, $\arcsin x$, $\operatorname{arctg} x$ и др. — не более 2 с.

Результат вычислений функций, помещаемый в регистр X и отображаемый на экране индикатора, может быть использован в качестве исходного данного для последующих вычислений.

Относительные погрешности значений функций, вычисляемых на ПМК, и допустимые пределы изменения их аргументов x приведены в табл. 2.2.

Как видно из таблицы, при вычислении значений функций из восьми знаков, отображаемых на индикаторе, во всех случаях не гарантируется точность и седьмого знака, а при вычислении квадратного корня — даже четвертого. В общем можно считать, что ПМК вычисляет значения функций с гарантированной точностью до шестого знака, кроме квадратного корня.

Упражнения

2.2.1. Вычислить значения функций:

1) $\sin 12^\circ =$	6) $\arcsin 3.45 =$	11) $0.941^2 =$	16) $\sqrt{-25} =$
2) $\cos 45^\circ =$	7) $\arccos -0.541 =$	12) $81.25^2 =$	17) $1/0.25 =$
3) $\tg 30^\circ =$	8) $\operatorname{arctg} 2345 =$	13) $10^{0.5} =$	18) $1/-10^9 =$
4) $\cos 10^{-11} =$	9) $e^{28.35} =$	14) $10^{-0.5} =$	20) $\lg 630.25 =$
5) $\sin -31^\circ =$	10) $e^{-0.458} =$	15) $\sqrt{625} =$	21) $\lg (-241.5) =$

П р и м е ч а н и е. Углы в примерах 6, 7, 8 определить в радианах и градусах.

2.2.2. Выполнить перевод значений угла «градусы, минуты, → градусы, доли градусов»:

1) $0^\circ 15.5' =$	5) $22^\circ 10' 30'' =$
2) $-30^\circ 12.54' =$	6) $1256^\circ 16.7' =$
3) $143^\circ 65' =$	7) $63^\circ 00.01' =$
4) $34^\circ 15' =$	8) $1^\circ 00.0001' =$

2.2.3. Выполнить перевод значений угла «градусы, доли градусов → градусы, минуты»:

1) $0.354^\circ =$	3) $16343.825^\circ =$
2) $-30.125^\circ =$	4) $0.000625^\circ =$

2.2.4. Выполнить перевод значений времени «часы, минуты, секунды → часы, доли часа»:

1) $6 \text{ ч } 32 \text{ мин } 18 \text{ с}$	3) $0 \text{ ч } 00 \text{ мин } 32 \text{ с} =$
2) $3444 \text{ ч } 12 \text{ мин } 44 \text{ с}$	4) $-124 \text{ ч } 01 \text{ мин } 0.05 \text{ с} =$

2.2.5. Выполнить перевод значений времени «часы, минуты, секунды → часы, доли часа»:

1) $11.725 \text{ ч} =$	3) $156.00001 \text{ ч} =$
2) $0.00625 \text{ ч} =$	4) $-12.5 \text{ ч} =$

2.3. ДВУХМЕСТНЫЕ ОПЕРАЦИИ

Операции, выполняемые над двумя числами operandами, называются двухместными. В ПМК предусмотрено выполнение следующих двухместных операций: арифметических (+, -, ×, ÷), логических (\vee , \wedge , \oplus , инв), возведение в степень (x^y) и нахождение максимального числа из двух чисел, находящихся в регистрах X

и Y. Основными, наиболее часто употребляемыми операциями являются арифметические и возведение в степень. Ограничимся рассмотрением только этих операций.

Для того чтобы ввести два числа в память ПМК и поместить одно из них в регистр X, а другое в регистр Y, достаточно набрать их одно за другим на клавиатуре, отделив ввод одного от ввода другого командой B↑, перемещающей только что введенное число из регистра X в регистр Y. Тогда первое набранное число будет помещено в регистр Y, а второе — в регистр X. Для упрощения написания команды B на пишущей машинке будем обозначать ее в дальнейшем, как B!.

Для хранения чисел при выполнении двухместной операции используются два регистра: Y — для первого числа операнда и X — для второго. Результат операции помещается в регистр X и отображается на экране индикатора. При этом старое содержимое регистра X (значение второго числа) стирается. Его, однако, можно восстановить, так как перед выполнением двухместной операции содержимое регистра X, отображаемое на экране индикатора, запоминается в регистре предыдущего результата — XI.

Условимся обозначать: (X) — содержимое регистра X, а (Y) — содержимое регистра Y. Содержимое регистров X и Y, которое будет иметь место до или после выполнения операции, условимся обозначать в скобках (X, Y), где на первом месте будет записано содержимое регистра X, а на втором — регистра Y.

Если в скобках второе число будет отсутствовать, то это означает, что содержимое регистра Y для нас безразлично и не используется.

Условимся кратко записывать:

Клавиша	Выполняемое действие
—	(Y) — (X) → X

Это означает, что при нажатии клавиши — (минус) из содержимого регистра Y вычитается содержимое регистра X, а результат операции (разность) помещается в регистр X и, следовательно, отображается на экране индикатора. Пусть, например, требуется сложить два числа 12 и 15. Тогда последовательность команд, реализующих эту операцию, будет выглядеть так:

00. 1 (1)	03. 1 (1, 12)
01. 2 (12)	04. 5 (15, 12)
02. B! (12, 12)	05. + (27)

ВЫДАТЬ 27.

Если использовать предписанное ВВЕСТИ, то последовательность команд, реализующих сложение этих же чисел, можно записать короче:

ВВЕСТИ 12 (12)	01. + (27)
00. B! (12, 12)	ВЫДАТЬ 27
ВВЕСТИ 15 (15, 12)	

Для еще более компактной записи условимся с помощью предписания ВВЕСТИ выполнять ввод в память ПМК не одного числа,

Таблица 2.3

Клавиши, используемые для выполнения двухместных арифметических операций

Обозначение клавиши	Выполняемые действия	Примечания
+	$(Y) + (X) \rightarrow X$	
-	$(Y) - (X) \rightarrow X$	
\times	$(Y) \times (X) \rightarrow X$	
\div	$(Y) : (X) \rightarrow X$	\div клавиша деления
x^y	$(X)^{(Y)} \rightarrow X$ при $(X) > 0$	Выполняется с помощью логарифмирования с точностью 10^{-6} . Время вычисления 3.5 с.
\leftrightarrow	$(X) \rightarrow Y$ $(Y) \rightarrow X$	Обмен содержимым между регистрами X и Y. Используется при необходимости поместить уменьшаемое или делимое из регистра X в регистр Y, или же прочитать содержимое регистра Y
$B \uparrow$ или $B!$	$(X) \rightarrow X$ $(X) \rightarrow Y$	Содержимое регистра X сохраняется Содержимое регистра Y заменяется содержимым регистров X Используется при вводе для разделения одного числа от другого

а нескольких (до четырех). При этом все числовые данные, перечисляемые в предписание ВВЕСТИ условимся отделять друг от друга символом ! (знак восклицания), предусматривающим нажатие клавиши B! после набора каждого числа. Тогда выполнение той же двухместной операции можно записать в виде команд:

ВВЕСТИ 12! 15(15, 12)
00. + (27)
ВЫДАТЬ 27

Двухместные арифметические операции, предусмотренные в ПМК, а также клавиши, с помощью которых они осуществляются, приведены в табл. 2.3.

Порядок выполнения двухместных операций. Для выполнения арифметических операций (+ — \times :) необходимо:

- 1) ввести первое число в регистр X;
- 2) поместить его в регистр Y, нажав клавишу B!
- 3) ввести второе число;
- 4) нажать клавишу, предписывающую выполнение соответствующей операции;
- 5) прочитать на экране индикатора результат операции (содержимое регистра X).

П р и м е ч а н и е. Если числа вводятся в регистры X и Y не путем набора их с пульта управления, а путем чтения их из регистров оперативной памяти,

то п. 2 при этом исключается. Перемещение первого введенного числа из регистра X в регистр Y производится автоматически без подачи команды B!, после подачи команды на ввод второго числа из регистра оперативной памяти в регистр X.

П р и м е р ы:

1. Сложение:

1) $A + B =$

ВВЕСТИ A! B (B, A)

00. + (A + B)

ВЫДАТЬ A + B

2. Вычитание:

1) $A - B =$

ВВЕСТИ A! B (B, A)

00. - (A - B)

ВЫДАТЬ A - B

3. Умножение:

1) $A \times B =$

ВВЕСТИ A! B (B, A)

00. $\times (A \times B)$

ВЫДАТЬ (A \times B)

4. Деление:

1) $A : B =$

ВВЕСТИ A! B (B, A)

00. $\div (A / B)$

ВЫДАТЬ A / B

2) $12 + 3 =$

ВВЕСТИ 12! 3 (3, 12)

00. + (15)

ВЫДАТЬ 15

2) $12 - 3 =$

ВВЕСТИ 12! 3 (3, 12)

00. - (9)

ВЫДАТЬ 9

2) $12 \times 3 =$

ВВЕСТИ 12! 3 (3, 12)

00. $\times (36)$

ВЫДАТЬ 36

2) $12 : 3 =$

ВВЕСТИ 12! 3 (3, 12)

00. $\div (4)$

ВЫДАТЬ 4

Особенности выполнения двухместной операции возведения в степень. Операция возведения числа x , находящегося в регистре X, в степень y (показатель степени при этом должен быть в регистре Y) осуществляется после нажатия клавиши X^y . Основание и показатель степени могут быть как целым числом, так и правильной или неправильной десятичной дробью.

Выполнение операции возведения в степень всегда производится с помощью логарифмирования. Отсюда следует, что она возможна только в том случае, когда основание степени — число положительное. Этим объясняется большое время выполнения операции (3,5 с) и пониженная точность вычислений (только до 6-го знака). Поэтому возведение числа в квадрат следует производить с помощью одноместной операции x^2 , а в куб — умножением: $x^2 \times x$. Время выполнения всех двухместных операций, за исключением возведения в степень, не более 0,5 с.

Для возведения числа A в степень B необходимо:

- 1) ввести в регистр X значение показателя степени B;
- 2) поместить показатель степени в регистр Y, нажав для этого клавишу B!;

3) ввести в регистр X основание степени A;

4) нажать клавишу FX^y;

5) прочитать результат на экране индикатора (в регистре X).

Более коротко последовательность действий пользователя при выполнении операции возведения в степень можно записать так:

ВВЕСТИ B! A (A, B)

00. F X^y (A^B)

ВЫДАТЬ A^B

Приимеры:

1) $21.7^{15.6} =$
ВВЕСТИ 15.6 ! 21.7 (21.7, 15.6)

00. FX^y (7.0594552 20)
ВЫДАТЬ 7.0594552 · 10²⁰

2) $4^{-0.2} =$

ВВЕСТИ —0.2 ! 4 (4, —2 —02)

00. X^y (7.5785828 —01)
ВЫДАТЬ 0.75785828

Некорректные операции. Двухместная операция считается некорректной при попытках:

деления на нуль, т. е. когда $(Y) : (X)$, если $(X) = 0$;
возведения в степень при отрицательном или нулевом значении основания степени, т. е. когда $(X)^{(y)}$ при $X \leq 0$. В последнем случае будет иметь место попытка вычисления логарифма отрицательного числа (или нуля), что невозможно;

при попытке выполнения операции, результат которой больше 9.9999999.10⁹⁹ (по абсолютной величине).

При попытке выполнения некорректной операции на экране индикатора будет высвечиваться сигнал ошибки ЕГГОГ.

Упражнения

2.3.1. Сложить следующие числа:

1) 228 + 301 =	5) 34.01 + 25 =
2) 3458 + 8809 =	6) 43.02 + 95.16 =
3) 95431185 + 34 =	7) 0.12345678 + 3.005 =
4) 24 · 10 ¹⁵ + 125 · 10 ¹⁶ =	8) 25.43 · 10 ⁻³ + 0.245 · 10 ⁻² =

2.3.2. Произвести вычитание чисел:

1) 310 — 241 =	5) 24 — 34.01 =
2) 241 — 310 =	6) 95.16 — 43.756 =
3) 12345678 — 81 =	7) 6.12345678 — 3.785 =
4) 26 · 10 ¹⁷ — 125 · 10 ¹⁵ =	8) 28.84 · 10 ⁻⁵ — 1.245 · 10 ⁻³ =

2.3.3. Умножить следующие числа:

1) 541 × 112 =	5) 64.25 × 11 =
2) 67829 × 24567 =	6) 84.001 × 112.7 =
3) 98765432 × 45678 =	7) 0.98765431 × 6.23 =
4) 41 · 10 ²⁷ × 24 · 10 ⁶⁵ =	8) 23.57 · 10 ⁻²⁵ × 6.23 =

2.3.4. Произвести деление следующих чисел:

1) 351 : 251 =	5) 84.11 : 21 =
2) 12567 : 23468 =	6) 65.167 : 84.3 =
3) 30384548 : 356 =	7) 19.100001 : 4.48 =
4) 286 · 10 ⁸ : 11 · 10 ¹² =	8) 39.11 · 10 ⁻⁶ · 0.59 · 10 ⁻³ =

2.3.5. Произвести возведение в степень:

1) 24 ³⁵ =	4) 3.6425 ^{10.5} =
2) 2 ¹² =	5) 12.46 ^{0.161} =
3) —3.6 ^{2.5} =	6) 224.171 ^{-8.61} =

2.3.6. Выполнить следующие действия:

1) sin 121° + cos 11° =	4) arcsin 0.87 — arccos 0.31 =
2) tg 0.561° — e ^{-31.5} =	5) arctg 1245 + 12.5 ² =
3) √[12.25 + 10 ^{-6.05}] =	6) 1/0.625 + π =
	7) lg 3.87 + ln 0.81 =

2.4. ПОСЛЕДОВАТЕЛЬНОСТЬ ОПЕРАЦИЙ

Наиболее часто с помощью ПМК приходится вычислять различные выражения, задаваемые расчетными формулами и значениями входящих в них переменных. Процесс вычисления таких выражений представляет собой некоторую последовательность выполняемых с помощью ПМК элементарных одноместных и двухместных операций, рассмотренных выше. При этом всегда стаются использовать результат предыдущей операции в качестве операнда (исходного данного) для предыдущей операции. Если это не удается, то промежуточные результаты приходится записывать отдельно и снова вводить их в ПМК, когда это необходимо.

Рассмотрим некоторые примеры выполнения многооперационных вычислений, часто встречающихся на практике, на которых покажем, как используются результаты предыдущих операций для выполнения последующих.

Суммирование последовательности чисел. Очень часто на практике приходится суммировать столбец или строку чисел. Это суммирование осуществляется путем накопления суммы в отдельном, предварительно обнуленном регистре памяти. Обычно в качестве такого регистра используется регистр Y, который перед началом суммирования обнуляется с помощью клавиш Cx и B!.

Пусть нам необходимо просуммировать столбец чисел:

15

—7

3.4

Процесс суммирования и записи промежуточных результатов в операционные регистры X и Y будет осуществляться путем выполнения следующих команд:

00. Cx (0)	ВВЕСТИ —7 (-7,15)
01. B! (0,0)	03. +(8)
ВВЕСТИ 15 (15, 0)	ВВЕСТИ 3.4 (3.4, 8)
02. +(15)	04. +(11.4)
	ВЫДАТЬ 11.4

Вычисление суммы обратных чисел. Пусть необходимо вычислить: $1/15 + 1/-7 + 1/3.4 =$.

Решение.

Для этого необходимо выполнить:

00. Cx (0)	05. +(—7.619047 —02)
01. B! (0, 0)	ВВЕСТИ 3.4 (3.4, —7.619047 —02)
ВВЕСТИ 15 (15, 0)	06. F 1/x (2.9411764 —01,
02. F 1/x (6.6666666 —02, 0)	—7.619047 —02)
03. +(6.6666666 —02)	07. +(2.1792717 —01)
ВВЕСТИ —7 (-7, 6.6666666 —02)	ВЫДАТЬ 0.21792717
04. F 1/x (—1.4285714 —01,	
6.6666666 —02)	

Вычисление суммы квадратов чисел. Пусть необходимо вычислить: $2,7^2 + (-16)^2 + 0,5^2 =$.

Решение:

00. Сх(0)
 01. В! (0,0)
 ВВЕСТИ 2.7 (2.7, 0)
 02. Fx² (7,29,0)
 03. +(7.29)
 ВВЕСТИ -16 (-16, 7.29)
 04. Fx² (256, 7.29)

05. +(263.29)
 ВВЕСТИ 0.5 (0.5, 263.29)
 06. Fx² (0.25,263.29)
 07. +(263.54)
 ВЫДАТЬ 263.54

Вычисление среднего арифметического. При вычислении среднего арифметического значения последовательности чисел производится их суммирование так, как это было показано выше, а затем полученный результат делится на число слагаемых. Пусть требуется вычислить:

$$M_{cp} = \frac{3.2 + 2.1 + 4.5}{3} =$$

Решение.

00. Сх(0)
 01. В! (0,0)
 ВВЕСТИ 3.2 (3.2, 0)
 02. +(3.2)
 ВВЕСТИ 2.1 (2.1, 3.2)
 03. +(5.3)
04. +(9.8)
 ВВЕСТИ 3 (3, 9.8)
 05. ÷(3.2666666)
 ВЫДАТЬ 3.2666666

Вычисление произведения, состоящего из нескольких сомножителей. В отличие от суммы слагаемых при вычислении произведения, состоящего из последовательности произведений, в накопительный регистр необходимо помещать не нуль, а единицу. Пусть необходимо вычислить: $4.1 \times 2.4 \times 1.5 =$.

Решение.

00. 1 (1)
 01. В! (1, 1)
 ВВЕСТИ 4.1 (4.1, 1)
 02. ×(4.1)
 ВВЕСТИ 2.4 (2.4, 4.1)
03. ×(9, 84)
 ВВЕСТИ 1.5 (1.5, 9.84)
 04. ×(14.76)
 ВЫДАТЬ 14.76

Упражнения

2.4.1. Произвести суммирование:

- | | | |
|-----------|---------|--------|
| 1) 3.8071 | 2) 63.1 | 3) 122 |
| 81.4900 | 27.2 | 34 |
| -9.1940 | 41.8 | 7 |
| 22.8153 | -11.4 | -164 |
| 7.1720 | 32.7 | 25 |
| <hr/> | <hr/> | <hr/> |
| 5.9012 | 17 | 17 |
| | <hr/> | <hr/> |
| | 985 | 985 |

2.4.3. Вычислить:

- 1) 5!=
 2) 10!=

2.4.4. Вычислить:

- 1) $\operatorname{tg} \frac{385 \cdot 9 \cdot 705}{17 \cdot 14}$
 3) $1/2.5 + 1/2.40 + 1/19.5 =$

- 2) $\frac{\ln (6618 + 222340)}{\lg 887} =$
 4) $0.13 + 1/265 + (1/1.8)^2 =$

2.4.2. Для упражнения 2.4.1 вычислить средние арифметические значения приведенных последовательностей чисел.

Раздел II

ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ В РУЧНОМ РЕЖИМЕ

Глава 3. ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПАМЯТИ ПМК

Одной из основных особенностей ПМК по сравнению с обычным МК является наличие в первом во много раз большего объема памяти, предназначено для записи и хранения числовых данных. Память данных, доступная пользователю ПМК, состоит из 15 регистров (в МК-54 их 14) адресуемой памяти и 4 регистров стековой памяти. Это позволяет запоминать до 19 значений исходных данных и промежуточных результатов не только при работе ПМК в автоматическом режиме, но и при выполнении вычислений вручную.

Адресуемая память ПМК (ОЗУ данных, см. рис. 1.1) позволяет обеспечить:

1. Однократную запись всех исходных данных в оперативную память ПМК, что дает возможность не прибегать к многократному их вводу в процессе решения задачи.

2. Запись в оперативную память ПМК всех промежуточных результатов, что дает возможность не прибегать к считыванию их с экрана индикатора, записи на бумаге и ручному вводу их на последующих этапах решения задачи.

Наличие стековой памяти, как будет показано ниже, дает возможность в значительной мере автоматизировать процесс записи и чтения промежуточных результатов и ускорить выполнение расчетов не только в автоматическом, но и в ручном режиме использования ПМК.

Наличие регистров памяти, в которых могут храниться значения переменных и выражений, дает возможность перейти от выполнения действий над конкретными числами к действиям над содержимым регистров, которое может быть для нас неизвестным. Это дает возможность составлять последовательность команд ПМК не только для решения задачи с каким-либо одним конкретным вариантом исходных данных, но и для решения той же задачи с любыми допустимыми исходными данными. А это создает предпосылки для автоматизации решения на ПМК задач, в которых используется регистровая память.

3.1. КОМАНДЫ ОБРАЩЕНИЯ К РЕГИСТРАМ АДРЕСУЕМОЙ ПАМЯТИ

В ПМК типа «Электроника» каждый из 15 регистров оперативного запоминающего устройства для данных (ОЗУ данных)

имеет свой номер (адрес). Чтобы подчеркнуть отличие этих регистров от регистров стековой памяти, их называют адресуемыми регистрами, а память, образованную совокупностью указанных регистров, — адресуемой памятью. Регистры обозначаются R0, R1, R2 ... R9, RA, RB, RC, RD, RE, где первый символ R обозначает регистр, а второй — номер регистра в шестнадцатеричной системе счисления (п. 1.2.2).

Запись числа в адресуемый регистр осуществляется с помощью клавиши $x \rightarrow \Pi$, предписывающей считывание числа из регистра X в регистр, адрес которого определяется клавишей, нажатой после клавиши $x \rightarrow \Pi$. Чтение числа из адресуемого регистра предписывается клавишей $\Pi \rightarrow x$. При этом адрес регистра определяется также нажатием последующей клавиши с соответствующим номером.

Следует отметить, что обозначение клавиш $x \rightarrow \Pi$ (запись в память) и $\Pi \rightarrow x$ (считывание из памяти) выбрано неудачно. При быстром вводе программ в ПМК эти клавиши легко путаются, что часто приводит к ошибкам в вычислениях. При работе на клавиатуре любого устройства часто встречаются симметричные ошибки, заключающиеся в том, что рядом расположенные клавиши даже с резко различающимися обозначениями невольно нажимаются в обратном порядке. Это знает любая машинистка. В данном случае рядом расположенные клавиши $x \rightarrow \Pi$ и $\Pi \rightarrow x$ имеют еще и симметричные обозначения. Более удобными были обозначения Π (в память) и ИП (из памяти) в ранее выпускавшихся ПМК БЭ-34 и принятые во всей литературе, выпущенной по ПМК к настоящему времени [2, 11–13, 16, 17, 19–21]. Для уменьшения вероятности ошибок указанного вида читателю рекомендуется делать акцент на правые части обозначений и называть про себя эти клавиши X (вместо $\Pi \rightarrow x$) и Π (вместо $x \rightarrow \Pi$), мысленно зачеркивая символы на клавишиах, помещенных слева. С этой целью и для упрощения записи первый символ будем обозначать малой буквой, а стрелку исключим. Тогда команду чтения из регистра Rn будем обозначать p_{Rn} , а команду записи в регистр Rn — x_{Pi} (вместо $\Pi \rightarrow x$ и $x \rightarrow \Pi$).

Запись числа в адресуемый регистр. Пусть число находится в регистре X и отображается на экране индикатора. Для того чтобы записать его в регистр с адресом n, необходимо последовательно нажать клавишу:

x_{Pi} , обозначающую «в память»;

n , соответствующую адресу регистра, в который необходимо записать число.

В частности, x_{PO} означает: записать число из регистра X в адресуемый регистр X, или сокращенно $(X) \rightarrow n$, где в данном случае скобки означают содержимое регистра X.

При записи числа в память старое содержимое адресуемого регистра стирается и на его место записывается содержимое регистра X, отображаемое на экране индикатора, которое сохра-

няется и после окончания записи. Клавиша $\times\text{P}$ может рассматриваться как префиксная. Запись числа, осуществляемая путем нажатия двух клавиш, считается одним шагом вычислений (одной командой).

Примеры:

1) Записать число 237 в регистр R0:
ВВЕСТИ 237 (237)

00. хП0 (237)
2) Записать число $6.02 \cdot 10^{23}$ в регистр RA:
ВВЕСТИ $6.02 \cdot 10^{23}$ (6.02_23)
00. хПА (6.02_23)

Если обозначение каждого из регистров R0—RE рассматривать как наименование (имя) некоторой переменной, то содержимое регистра будет представлять собой значение этой переменной. Тогда операцию записи числа в адресуемый регистр можно рассматривать как присваивание переменной ее конкретного значения. Операцию присваивания обычно записывают как $B := A$, или проще: $B = A$. Такая запись не означает, что $B = A$. Напротив, как правило, она означает, что B не равно A до выполнения операции. И лишь после ее выполнения B приравнивается A , т. е. конкретное значение A записывается в регистр, отведенный для хранения значений переменной B . Условимся операцию присваивания записывать в виде:

регистр = выражение,

понимая число или имя переменной как частный случай выражения. Рассмотрим примеры операций присваивания и реализацию их с помощью команд ПМК:

Операция присваивания
R0=237

RA= $6.02 \cdot 10^{23}$

R9=3.14

Команды ПМК
ВВЕСТИ 237
00. хП0
ВВЕСТИ $6.02 \cdot 10^{23}$
01. хПА
ВВЕСТИ 3.14
02. хП9

Если перед решением задачи на ПМК ввод исходных данных осуществляется не в операционные регистры X и Y, а в адресуемые R0—RE, то операции ввода и присваивания можно записать с помощью одного предписания:

- 1) ВВЕСТИ R0=237 (237)
- 2) ВВЕСТИ RA= $6.02 \cdot 10^{23}$ (6.02_23)
- 3) ВВЕСТИ R9=—3.14 (—3.14)

Чтение числа из адресуемого регистра. Пусть в регистре с адресом Rn записано число A . Пусть в это время на экране индикатора отображается число B , хранящееся в регистре X. Тогда при чтении содержимого регистра n (числа A) последнее будет записано в регистр X и отображено на экране. Старое содержимое ре-

гистра X (число B) будет перенесено при этом в регистр Y. Содержимое регистра n (число A) останется после чтения неизменным.

Чтение числа из адресуемого регистра производится путем последовательного нажатия клавиши pX («из памяти»), а затем клавиши, соответствующей номеру регистра, содержимое которого необходимо прочитать. Например, для чтения числа из регистра R0 необходимо нажать клавиши pX и 0, а для чтения из регистра B — клавиши pX и B, в результате чего в регистре X и на экране индикатора появится содержимое соответствующего регистра.

Клавишу pX также можно рассматривать как префиксную. Чтение одного числа из адресуемого регистра в регистр B считается одним шагом (одной командой). Поэтому в дальнейшем будем записывать команды чтения слитно pX0, pXB.

Примеры:

Пусть в регистрах адресуемой памяти ранее были записаны следующие числа:

R0 = 237; RA = $6.02 \cdot 10^{23}$; R9 = —3.14; RC = $-8.53 \cdot 10^{-23}$; R1 = 0.086.

Требуется: Прочитать последовательно содержимое регистров R0, RA, R9, RC, R1 и посмотреть, какие числа в результате чтения будут записаны в регистры X и Y.

Решение:

00. pX0 (237);
01. pXA (6.02_23, 237);
02. pX9 (—3.14, 6.02_23);
03. pXC (—8.53_23, —3.14);
04. pX1 (8.6_02, —8.53_23).

Далее, требуется переписать числа из регистров, в которых они были ранее записаны, в другие, а именно:

R2=R0; RB=RA; R8=R9; R0=RC; R5=R1.

Решение:

00. pX0 (237)
01. pX2 (R2=237)
02. pxA (6.02_23, 237)
03. xPB (RB=6.02_23)
04. pX9 (—3.14, 6.02_23)
05. xP8 (R8=—3.14)
06. pXC (—8.53_23, —3.14)
07. xP0 (R0=—8.53_23)
08. pX1 (8.6_02, —8.53_23)
09. xP5 (R5=8.6_02)

Как видно из приведенных примеров, при чтении числа из адресуемого регистра нет необходимости в использовании клавиши B!, перемещающей содержимое регистра X в регистр Y, поскольку такое перемещение обеспечивается автоматически в процессе выполнения операции чтения.

3.2. ИСПОЛЬЗОВАНИЕ АДРЕСУЕМОЙ ПАМЯТИ ДЛЯ ВЫЧИСЛЕНИЯ СЛОЖНЫХ ВЫРАЖЕНИЙ

Адресуемая память ПМК может быть эффективно использована при вычислении сложных выражений для записи промежуточных результатов. Рассмотрим вычисление площади треугольника по формуле Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a + b + c)/2$, на двух примерах: в первом из них расчеты будут производиться без использования адресуемой памяти (промежуточные результаты будут записываться в дополнительных графах

Таблица 3.1

Расчет площади треугольника по формуле Герона
без использования адресуемой памяти

$a, \text{ м}$	$b, \text{ м}$	$c, \text{ м}$	p	$p - a$	$p - b$	$p - c$	$S, \text{ м}^2$
0.25	0.5	0.6	0.675	0.425	0.175	0.075	6.1361505—02
0.50	1.0	1.2	1.35	0.850	0.35	0.15	2.4544602—01
1.00	2.00	2.5	2.75	1.75	0.75	0.25	9.4991775—01

П р и м е ч а н и е. Следует заметить, что далеко не всегда целесообразно записывать в таблицу результаты с точностью, выдаваемой ПМК. Если, например, исходные данные имеют точность два знака после точки, то полученные результаты следует округлить также до двух знаков. Однако здесь и в дальнейшем мы этого делать не будем, чтобы дать возможность читателю сверить результаты, получаемые на экране индикатора, с результатами, приводимыми в данном пособии.

таблицы), а во втором — для записи промежуточных результатов будут использованы регистры адресуемой памяти.

П р и м е р:

1. Расчет площади треугольника без использования адресуемой памяти ПМК. Вычисления производятся с помощью таблицы 3.1, в каждой строке которой записываются исходные данные, а также промежуточные и окончательные результаты расчета площади одного треугольника.

Расчеты в табл. 3.1 выполняются в следующей последовательности:

1. Вычисление p :
 ВВЕСТИ $a \mid b (b, a)$
 00. $+(a+b)$
 ВВЕСТИ $c (c, a+b)$
 01. $+(a+b+c)$
 02. $2 (2, a+b+c)$
 03. $\div ((a+b+c)/2)$
 ВЫДАТЬ p
 2. Вычисление $p - a$:
 ВВЕСТИ $p \mid a (a, p)$
 00. $-(p-a)$
 ВЫДАТЬ $p - a$
 3. Вычисление $p - b$:
 ВВЕСТИ $p \mid b (b, p)$
 00. $-(p-b)$
 ВЫДАТЬ $p - b$

4. Вычисление $p - c$:
 ВВЕСТИ $p \mid c (c, p)$
 00. $-(p-c)$
 ВЫДАТЬ $p - c$
 5. Вычисление S :
 ВВЕСТИ $p \mid p - a (p-a, p)$
 00. $\times (p(p-a))$
 ВВЕСТИ $p - b (p-b, p(p-a))$
 01. $\times (p(p-a)(p-b))$
 ВВЕСТИ $p - c (p-c, p(p-a)(p-b))$
 02. $\times (p(p-a)(p-b)(p-c))$
 03. $F V (p(p-a)(p-b)(p-c))$
 ВЫДАТЬ S

Без использования адресуемой памяти промежуточные результаты приходится считывать с экрана индикатора и записывать вручную в соответствующую графу таблицы, что требует определенных затрат времени. При этом возможно появление ошибок во время считывания и записи чисел (особенно многоразрядных). Использование регистров адресуемой памяти для хранения исходных данных и промежуточных результатов регистры адресуемой памяти ускоряет процесс вычислений и повышает их надежность. При этом объем данных, подлежащих занесению в таблицу, будет значительно сокращен.

2. Расчет площади треугольника по формуле Герона с использованием адресуемой памяти ПМК (табл. 3.2).

Использование адресуемой памяти для выполнения вычислений предполагает необходимость предварительного распределения ее регистров для хранения значений исходных данных и промежуточных результатов. Это распределение записывается перед командами ПМК в следующем виде: $R0=a$; $R1=b$; $R2=c$; $R3=p$; $R4=p-a$; $R5=p-b$; $R6=p-c$ (здесь и в дальнейшем все предписания условимся отделять друг от друга запятой, если они записаны в одну строку). Дальнейшую последовательность действий при расчете площади треугольника (одной строки таблицы) можно записать следующим образом:

ВВЕСТИ $R0=a, R1=b, R2=c$

00. $\pi X0 (a)$
 01. $\pi X1 (b, a)$
 02. $+(a+b)$
 03. $\pi X2 (c, a+b)$
 04. $+(a+b+c)$
 05. $2 (2, a+b+c)$
 06. $\div ((a+b+c)/2)$
 07. $x\pi 3 (R3 = p)$ запись $p \rightarrow R3$;
 08. $\pi X0 (a, p)$ } вычисление $p - a$;
 09. $-(p-a)$ } вычисление $p - a$;
 10. $x\pi 4 (R4 = p - a)$ запись $p - a \rightarrow R4$;
 11. $\pi X3 (p, p - a)$ чтение p ;
 12. $\pi X1 (b, p)$ } вычисление $p - b$;
 13. $-(p-b)$ } вычисление $p - b$;
 14. $x\pi 5 (R5 = p - b)$ запись $p - b \rightarrow R5$;
 15. $\pi X3 (p, p - b)$
 16. $\pi X2 (c, p)$ } вычисление $p - c$;
 17. $-(p-c)$ } вычисление $p - c$;
 18. $x\pi 6 (R6 = p - c)$ запись $p - c \rightarrow R6$;
 19. $\pi X3 (p)$
 20. $\pi X4 (p - a, p)$
 21. $\times (p(p-a))$
 22. $\pi X5 (p - b, p(p-a))$
 23. $\times (p(p-a)(p-b))$
 24. $\pi X6 (p - c, p)(p-a)(p-b)(p-c))$
 25. $\times (p(p-a)(p-b)(p-c))$
 26. $F V (s)$
 ВЫДАТЬ S

вычисление $p = (a + b + c)$;

вычисление S .

Таблица 3.2

$a, \text{ м}$	$b, \text{ м}$	$c, \text{ м}$	$S, \text{ м}^2$
0.25	0.50	0.60	6.1361505—02
0.50	1.00	1.20	2.4544602—01
1.00	2.00	2.50	9.4991775—01

Приведенная последовательность команд ПМК остается одной и той же для любых допустимых значений исходных данных a, b, c . Такую последовательность команд, не привязанную к конкретным исходным данным, принято называть программой решения задачи на ПМК. Ее выполнение может быть осуществлено пользователем как путем последовательного нажатия всех клавиш, соответствующих командам программы, так и автоматически, после ввода ее в программную память ПМК. Вопросы автоматического выполнения программ будут рассмотрены в последующих главах.

Сравнение табл. 3.1 и 3.2 показывает, что использование адресуемой памяти существенно облегчает действия пользователя ПМК при выполнении вычислений. Расчеты по табл. 3.1 без использования адресуемой памяти более наглядны и более понятны пользователю, хотя и более трудоемки. При использовании табл. 3.2 не надо записывать промежуточные результаты и считывать их с экрана индикатора, что повышает надежность расчета. Приведенная выше программа, как это видно из ее текста, ориентирована не столько на выполнение ее человеком, сколько на выполнение ее микрокалькулятором. Поэтому она должна быть предварительно составлена и записана на бумаге до выполнения вычислений. Дальнейшие действия пользователя по вводу команд программы и выполнению вычислений осуществляются чисто механически без выявления смысла каждой команды.

Другим недостатком вычислений с использованием адресуемой памяти является наличие большого числа команд программы, связанных с записью и чтением промежуточных результатов. В следующем параграфе будет показано, что процесс запоминания и чтения промежуточных результатов можно в значительной мере автоматизировать путем использования магазинной или стековой памяти.

Упражнения

3.2.1. Вычислить следующие выражения, используя при этом запись промежуточных результатов в регистрах адресуемой памяти:

$$1) T = \frac{24.3 \cdot 2.6 + 16.8 \cdot 3.3}{0.22 \cdot 5.65 + 8.58 \cdot 0.073} =$$

$$2) M = 20^6 + \sqrt[3]{14964} - 16^6 =$$

П р и м е ч а н и е. $\sqrt[3]{...}$ вычисляется как $(...)^{1/3}$.

$$3) T = \frac{343 \cdot 458 + 507 \cdot 985 + 387 \cdot 174}{114 + 234 + 859 + 484} =$$

3.2.2. Дано:

$$P = \sin^4 \varphi + \cos^4 \varphi - \frac{1 - \operatorname{tg}^2 \varphi}{\operatorname{tg}^2 \varphi}.$$

Требуется:

1) Составить последовательность команд для вычисления P при любых значениях φ .

2) Вычислить значение P при $\varphi = 72^\circ$.

3.2.3. Вычислить выражение с использованием адресуемой памяти (при $\operatorname{arctg} 1.2 = 2$ радианах):

$$P = \left(\lg \frac{(1/\sqrt{3}) + 3\sqrt{5} - \operatorname{arctg} 1.2) \cdot 10^{-3}}{\ln 2 \cdot \ln 3 + \ln 4 \cdot \ln 5} - 2.2^9 \right) + 2.2^9.$$

3.2.4. Произвести расчет коэффициента усиления параболической антенны в децибелах (dB) по формуле

$$G = 10 \lg \left(K \left(\frac{\pi D}{\lambda} \right)^2 \right),$$

где $K = 0.5 \div 0.8$ — коэффициент использования поверхности раскрытия антенны; D — диаметр раскрытия антенны, м; λ — длина волны, м.

1) Составить программу вычисления коэффициента усиления G .

2) Рассчитать значения коэффициента усиления для $K = 0.6$, $D = 0.8$ м и $\lambda = 0.03$ м.

3.3. ПОНЯТИЕ О БЕССКОБОЧНОЙ ЗАПИСИ ВЫРАЖЕНИЙ

Для выполнения расчетов на ПМК наиболее удобными являются выражения, вычисление которых сводится к такой линейной последовательности операций, в которой результат каждой предыдущей операции служит операндом (исходным данным) для последующей. Очевидно, что выражения, удобные для вычисления на ПМК, не должны содержать ни скобок, ни операций различного старшинства, изменяющих естественный порядок выполнения операций. Так как в общепринятой записи выражений это, как правило, не соблюдается, то возникает вопрос, нельзя ли общепринятую запись арифметических выражений преобразовать к такому виду, где не будет скобок и все операции будут одинакового старшинства.

Оказывается, можно. Такую запись выражений, которая получила название польской инверсной (обратной) записи, предложил польский математик Лукашевич. В этой записи операнды числа и операторы (предписания) располагаются таким образом, что исключается необходимость в применении скобок. Причем всегда сохраняется линейный порядок их выполнения. Поэтому этот тип записи часто называют бесскобочной обратной записью, в которой операции (+ — × ÷) записывают только после операндов (переменных или числовых констант). Существует также прямая бесскобочная запись, в которой знаки операций записывают перед записью операндов. Поскольку ПМК рассчитан на вычисление выражений в обратной бесскобочной записи, то прямую бесскобочную запись рассматривать не будем и, говоря о бесскобочной записи, будем в дальнейшем подразумевать первую.

Рассмотрим примеры представления обычных выражений в бесскобочной записи [12]:

Обычная запись

$$(a + b)(c - d)$$

$$\frac{ad - bc}{uz - vy}$$

$$(\ln(x + \sqrt{yz - \sin u}))$$

Бесскобочная запись

$$abcX +$$

$$ab + cd - x$$

$$ad \times bc \times - uz \times vy - :$$

$$yz \times u \sin - \sqrt{x} + \ln$$

На первый взгляд бесскобочная запись кажется настолько не привычной, что невольно вызывает реакцию отторжения. Но если несколько изменить форму приведенной выше бесскобочкой записи, не меняя ее сути (а суть ее заключается в том, что это способ вычисления обычных выражений), то картина меняется. Бесскобочная запись в той форме, в которой она применяется для расчетов на ПМК, практически никаких трудностей в ее восприятии не вызывает.

Представим себе магазин стрелкового оружия, который вместо патронов поочередно заполняется числами — значениями переменных или константами. При выполнении одноместной операции (вычисления функции одного аргумента) число на самом верхнем уровне (первом, соответствующем тому уровню, с которого патрон подается в патронник или заряжается магазин) преобразуется в соответствующее значение функции. При этом никакого перемещения в магазине с уровня на уровень не происходит. Двухместные операции ($+ - \times : \div$) выполняются над числами, размещенными на первом (самом верхнем) и втором уровнях. При этом в случае вычитания и деления уменьшающее и делимое помещаются на первый уровень, а все остальные числа перемещаются в магазине снизу вверх на один уровень.

Любое вводимое число заполняет самый верхний первый уровень. При этом все остальные числа, хранящиеся в магазине, перемещаются вниз на один уровень.

Для того чтобы определить значение любого выражения, весь процесс его вычисления необходимо подразделить на элементарные шаги. Элементарным шагом процесса вычислений будем считать либо ввод (запоминание, запись на бумаге или в памяти ПМК) значения переменной, либо выполнение одной операции (одноместной или двухместной). Условимся шаги процесса вычислений обозначать номерами от 00 до pn , причем после номера каждого шага ставить точку.

Например, пусть необходимо вычислить площадь параллелограмма по формуле $S = ab \sin C$, где C — угол между сторонами a и b . При наличии магазинной памяти процесс вычислений будет выполняться следующим образом:

- обычная запись: $ab \sin C$;
- бесскобочная запись: $abC \sin X$.

Состояние магазина при выполнении вычислений:

a	b	C	\sin	\times	\times
a	b	C	$\sin C$	$b \sin C$	$ab \sin C$
			b	a	a
			a		

При перемещении числа с самого нижнего уровня на один уровень вверх содержимое этого нижнего уровня сохраняется.

Процесс вычисления бесскобочного выражения можно представить так, как показано в табл. 3.3.

Таблица 3.3

Процесс вычисления площади параллелограмма по формуле с использованием магазинной памяти

Шаг вычислений	Состояние магазина (уровень)		
	первый	второй	третий
00. Ввод a	a		
01. Ввод b	b	a	
02. Ввод C	C	b	a
03. Вычисление \sin	$\sin C$	b	a
04. Умножение (\times)	$b \sin C$	a	a
05. Умножение (\times)	$ab \sin C$	a	a

В дальнейшем процесс вычислений с использованием магазинной памяти будем представлять без таблиц в форме, наиболее удобной для расчетов на ПМК.

Значения переменных в магазине, которые являются результатом выполнения шага вычислений, условимся помещать в круглых скобках правее символов операции, выполняемой на данном шаге. В скобках будем помещать список данных, хранящихся в магазине после выполнения шага вычислений. Элементы списка отделяются друг от друга запятыми. Ими могут быть переменные величины, выражения (но всегда подразумевается, что хранятся в магазине конкретные значения) или числовые константы. При этом первый элемент списка будет характеризовать содержимое первого уровня магазина, второй элемент — второго уровня и т. д. Ввод значений переменных условимся отображать путем записи соответствующими знаками операции, как это показано в табл. 3.3.

Покажем теперь, как преобразовать обычные выражения к виду, удобному для вычисления на ПМК, используя приведенные выше примеры обычной и бесскобочкой запись.

Примеры:

1. Обычная запись: $a + bc$.

Запись, удобная для вычисления на ПМК:

- | | |
|-------------------|----------------------|
| 00. $a (a)$ | 03. $\times (bc, a)$ |
| 01. $b (b, a)$ | 04. $+$ ($a + bc$) |
| 02. $c (c, b, a)$ | |

Легко видеть, что эта запись представляет собой не что иное, как видоизмененную форму бесскобочкой записи. Прочитав последовательность шагов (без учета их номеров и содержимого магазина после каждого шага) сверху вниз, получим $abc +$, т. е. бесскобочную запись обычного выражения $a + bc$.

2. Обычная запись: $(a + b)(c - d)$.

Запись, удобная для вычислений на ПМК:

00. $a(a)$ 04. $d(d, c, a+b)$
01. $b(b, a)$ 05. $-c-d, a+b$
02. $+(a+b)$ 06. $\times(a+b)(c-d)$
03. $c(c, a+b)$

Бесскобочная запись: $ab + cd - \times$.

3. Обычная запись: $\frac{ad-bc}{uz-vy}$.

Запись, удобная для вычислений на ПМК:

00. $a(a)$ 08. $z(z, u, ad-bc)$
01. $d(d, a)$ 09. $\times(uz, ad-bc)$
02. $\times(ad)$ 10. $v(v, uz, ad-bc)$
03. $b(b, ad)$ 11. $y(y, v, uz, ad-bc)$
04. $c(c, b, ad)$ 12. $\times(vy, uz, ad-bc)$
05. $\times(bc, ad)$ 13. $- (uz-vy, ad-bc)$
06. $- (ad-bc)$ 14. $\div ((ad-bc)/(uz-vy))$
07. $u(u, ad-bc)$

Бесскобочная запись: $ad \times bc \times -z \times vy \times -$.

4. Обычная запись: $\ln(x + \sqrt{yz - \sin u})$.

Запись, удобная для вычислений на ПМК:

00. $y(y)$ 05. $-(yz - \sin u)$
01. $z(z, y)$ 06. $\sqrt{(yz - \sin u)}$
02. $\times(yz)$ 97. $\times(x, \sqrt{yz - \sin u})$
03. $u(u, yz)$ 08. $+(x + \sqrt{yz - \sin u})$
04. $\sin(\sin u, yz)$ 09. $\ln(\ln(x + \sqrt{yz - \sin u}))$

Бесскобочная запись: $yz \times u \sin -V \times + \ln$.

Приведенные примеры подтверждают, что для вычисления выражений, представленных в бесскобочной записи, необходима память, организованная по принципу магазина стрелкового оружия. Такую память принято называть стековой, а сам магазин — стеком (от английского слова *stack*, что означает набор, штабель, куча).

При достаточном количестве регистров стека можно любое обычное выражение преобразовать в бесскобочное. Для многих задач достаточно иметь в стеке два регистра. Во всех предыдущих примерах, начиная с приведенных в п. 2.4, мы, не говоря об этом, фактически вычисляли бесскобочные выражения, используя только два регистра стека — операционные регистры X и Y, которые работают по принципу магазинной памяти. При этом преобразование обычных выражений в бесскобочные делалось в уме и никаких особых трудностей не вызывало.

Для вычисления бесскобочных выражений, рассмотренных в примерах настоящего параграфа, достаточно иметь в стеке от 2 до 4 регистров. Стек большинства ПМК имеет 4 регистра. Если емкость стека окажется недостаточной, то переходят к использованию адресуемой памяти.

Упражнения

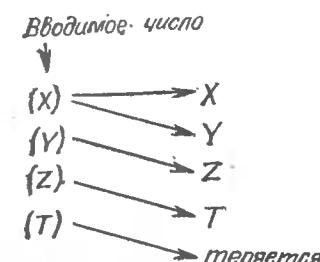
3.3.1. Преобразовать выражения в обычной записи в бесскобочную, удобную для расчетов на ПМК:

- 1) $2v \cos \alpha$;
2) $(1 + 0.6 \cos(2.4t + 5\pi/18)) \cos 4t$.

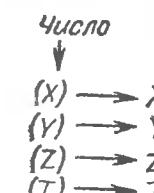
3.4. СТЕКОВАЯ ПАМЯТЬ ПМК. ЗАПОЛНЕНИЕ И ВЫБОРКА ЧИСЕЛ ИЗ СТЕКА

В ПМК типа «Электроника» стековая память (или просто стек) состоит из четырех регистров, обозначаемых X, Y, Z, T. Регистры X и Y являются операционными. В указанные регистры помещаются числа (операнды), над которыми требуется выполнить предписываемые операции. Регистр X является входным регистром стека, т. е. числа всегда первоначально вводятся в этот регистр.

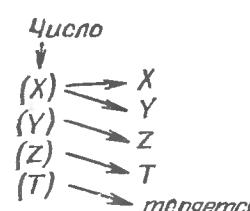
Перемещение при выполнении различных команд ПМК можно изобразить на следующих схемах:



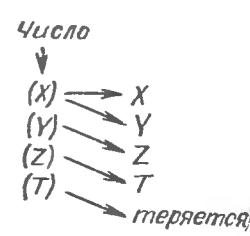
1. Ввод чисел с пульта ПМК после выполнения любой операции (нажатия любых клавиш, кроме клавиш набора чисел и клавиш управления работой программы).



2. Ввод следующего числа с пульта ПМК после ввода предыдущего и нажатия клавиши B!

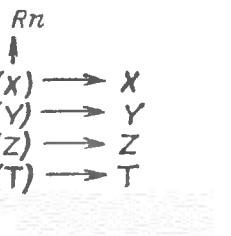


3. После окончания ввода очередного числа и нажатия клавиши B!

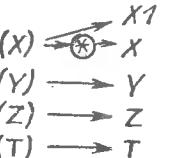


4. Ввод числа из регистра адресуемой памяти (команда nXn , где n — номер (адрес) регистра).

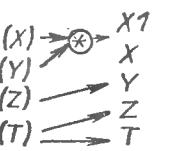
5. Считывание числа из регистра X в регистр адресуемой памяти (команда $xPln$).



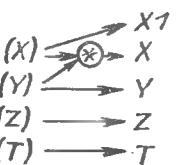
6. Выполнение одноместной операции команды $F \sin$, $F \cos$, $F \tg$, Fx^2 , $F\pi$, $/-$, $F \sin^{-1}$, $F \cos^{-1}$, $F \tg^{-1}$, $F \ln/x$, FV , Fe^x , $F \lg$, $F \ln$, $F10^x$ и др.
Результат \oplus — в регистр X.



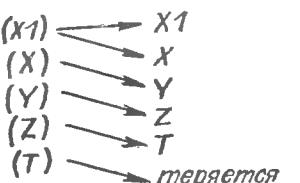
7. Выполнение двухместной операции команды $+ - X$:
Результат \otimes — в регистр X.



8. Выполнение операции возведения в степень (команда FXY).



9. Чтение содержимого регистра — результата предыдущего действия $X1$ (команда FBx).



Условимся обозначать состояние стека с помощью последовательности четырех чисел имен переменных, выражений, помещенных в скобки (...) и отделяемых друг от друга запятыми, например (a, b, c, d) . При этом первое число будет обозначать содержимое регистра X, второе — Y, третье — Z, четвертое — T. Если в скобках будет помещено меньше четырех чисел переменных величин, выражений, то это будет означать, что в данном случае для решаемой задачи безразлично, что находится в регистрах, содержащих которых не приведено.

3.4.1. СОСТОЯНИЕ СТЕККОВОЙ ПАМЯТИ ПОСЛЕ ВЫПОЛНЕНИЯ РАЗЛИЧНЫХ ОПЕРАЦИЙ

Ввод числа с клавиатуры и запись его в стек. Пусть состояние стека до ввода в него числа будет $(0, 0, 0, 0)$, т. е. все регистры стека сброшены (обнулены). Такое состояние всегда имеет место после включения ПМК. Если же на калькуляторе ранее производились какие-либо вычисления, то обнуление стека может быть достигнуто путем выполнения следующей последовательности команд:

00. Cx (0) 01. B! (0, 0) 02. B ! (0, 0, 0) 03. B ! (0, 0, 0, 0)

Рассмотрим различные варианты ввода с клавиатуры ПМК последовательности чисел 1, 2, 3, 4, 5, и их размещение в стеке в зависимости от этих вариантов.

Вариант 1. Нажмем последовательно клавиши 1, 2, 3, 4, 5, не фиксируя ничем окончание ввода каждого числа. Тогда в регистре X окажется введенным число 12345, а в остальных регистрах стека будут нули.

00. 1 (1, 0, 0, 0)	03. 3 (123, 0, 0, 0)	05. 5 (12345, 0, 0, 0)
01. 2 (12, 0, 0, 0)	04. 4 (1234, 0, 0, 0)	

Вариант 2. Окончание ввода каждого числа пусть будет зафиксировано началом какой-либо операции. В качестве такой операции можно использовать команду КНОП (нет операции). Тогда состояние стека будет последовательно изменяться следующим образом:

00. 1 (1, 0, 0, 0)	03. КНОП (2, 1, 0, 0)	06. 4 (4, 3, 2, 1)
01. КНОП (1, 0, 0, 0)	04. 3 (3, 2, 1, 0)	07. КНОП (4, 3, 2, 1)
03. 2 (2, 1, 0, 0)	06. КНОП (3, 2, 1, 0)	08. 5 (5, 4, 3, 2)

Если ввод числа с клавиатуры завершается командой КНОП, то по окончании ввода каждого очередного числа происходит автоматическое перемещение числа в стеке от регистра X по направлению к регистру T.

Вариант 3. Окончание ввода каждого числа фиксируется командой B!

00. 1 (1, 0, 0, 0)	03. B! (2, 2, 1, 0)	06. 4 (4, 3, 2, 1)
01. B! (1, 1, 0, 0)	04. 3 (3, 2, 1, 0)	07. B! (4, 4, 3, 2)
02. 2 (2, 1, 0, 0)	05. B! (3, 3, 2, 1)	08. 5 (5, 4, 3, 2)

Если ввод числа не заканчивается командой $B!$, то вводимое число записывается в регистр X без перемещения чисел в стеке (до нажатия клавиши $B!$). После ее нажатия осуществляется перемещение чисел в стеке по направлению к регистру T . При этом в регистре T сохраняется только что записанное число.

Чтение чисел из адресуемой памяти. Пусть в адресуемых регистрах памяти $R1-R5$ хранится последовательность чисел 1, 2, 3, 4, 5. Прочитаем последовательно числа 1, 2, 3, 4, 5 из регистров $R1, R2, R3, R4, R5$ соответственно и посмотрим, как будет при этом изменяться состояние стековой памяти:

00. $\text{p}X1\ (1)$	02. $\text{p}X3\ (3, 2, 1)$	04. $\text{p}X5\ (5, 4, 3, 2)$
01. $\text{p}X1\ (1)$	03. $\text{p}X4\ (4, 3, 2, 1)$	

Перемещение чисел в стеке при чтении числа из адресуемой памяти происходит автоматически от регистра X к регистру T (точно так же, как и при вводе чисел с клавиатуры, завершающимся командой KНОП . См. вариант 2).

При записи числа в адресуемый регистр из регистра X перемещение чисел в стеке не происходит.

Выполнение одноместных операций. При выполнении одноместных операций (вычисление функции одного аргумента) перемещение чисел в регистрах стека не происходит. На место значения аргумента x в регистре X записывается значение функции.

Например:

Начальное состояние стека (10, 20, 30, 40):

00. $F1/x\ (1 \rightarrow 01, 20, 30, 40)$
01. $F1/x\ (10, 20, 30, 40)$
02. $F10^x\ (1 \rightarrow 10, 20, 30, 40)$
03. $FV\ (100000, 20, 30, 40)$
04. $F\lg\ (5.0000002, 20, 30, 40)$

Выполнение двухместных операций. Пусть начальное состояние стека будет (1, 2, 3, 4). В случае выполнения двухместных операций действия будут производиться над числами, хранящимися в операционных регистрах X и Y , причем после выполнения операции результат будет помещаться в регистр X , а все числа, хранящиеся в других регистрах стека, будут перемещаться в сторону регистра X на одну позицию. Рассмотрим на примерах, как будет изменяться состояние стека после выполнения двухместных операций.

Примеры:

1. Получить сумму чисел, хранящихся в стеке.
Начальное состояние стека — (1, 2, 3, 4):

00. $+(3, 3, 4, 4)$
01. $+(6, 4, 4, 4)$
03. $+(10, 4, 4, 4)$

2. Вычислить произведение чисел, записанных в регистрах стека.
Начальное состояние стека — (1, 2, 3, 4):

00. $\times(2, 3, 4, 4)$
01. $\times(6, 4, 4, 4)$
02. $\times(24, 4, 4, 4)$

После выполнения двухместных арифметических операций ($+\ - \times \div$) происходит перемещение чисел в стеке от регистра T к регистру X . При этом содержимое регистра T сохраняется, а значения операндов, над которыми производятся арифметические действия, теряются. Прежнее содержимое регистра X (значение второго операнда) можно восстановить, прочитав его в регистре $X1$ (регистр предыдущего результата).

Выполнение операции возвведения в степень (команда FX^y). Пусть начальное состояние стека будет (4, 3, 2, 1). В команде возвведения в степень FX^y в качестве основания степени берется содержимое регистра X , а в качестве показателя степени — содержимое регистра Y . Рассмотрим, что будет выполнено по команде FX^y и как при этом будут перемещаться числа в стеке.

Начальное состояние стека — (4, 3, 2, 1):

00. $FX^y\ (64.000004, 3, 2, 1)$
01. $FX^y\ (262143.91, 3, 2, 1)$
02. $FX^y\ (1.8014373 \rightarrow 16.3, 2, 1)$
03. $FX^y\ (5.8459819 \rightarrow 48, 3, 2, 1)$

Как видно из этого примера, при операции возвведения в степень перемещения чисел в стеке не происходит. Изменяется лишь состояние регистра X : вместо основания степени в него помещается результат возвведения в степень, а значение показателя степени в регистре Y сохраняется без изменений.

Отсутствие перемещения чисел в стеке при выполнении рассматриваемой операции, а главное, сохранение показателя степени в регистре Y , создает значительные неудобства при использовании стековой памяти. Сохранение в стеке значения показателя степени играет примерно ту же роль, что и очередной заедающий патрон при подготовке выстрела. Если значение показателя степени не исключить из стека, то попытка использовать результат предыдущего действия, как и в обычной двухместной операции, приводит к ошибке при вычислениях. Поэтому после выполнения операции X^y необходимо либо очистить стек от показателя степени, либо отказаться от дальнейшего использования регистров X и Y и регистров оперативной памяти.

Учитывая, что $X^y = 10^{y \lg x}$, можно вычислить X^y не с помощью команды FX^y , а используя команды $F\lg$, умножения \times и $F10^x$. Пусть исходное состояние стека будет такое же, как и в предыдущем примере.

Начальное состояние стека — (4, 3, 2, 1):

00. $F\lg\ (6.0206 \rightarrow 01, 3, 2, 1)$
01. $\times(1.80618, 2, 1, 1)$
02. $F10^x\ (64.000004, 2, 1, 1)$

Сравнивая состояние стека с полученным в предыдущем примере (возведение в степень с помощью команды FX^y), можно прийти к выводу, что использование указанных трех команд вместо команды FX^y не изменяет обычного порядка перемещения чисел в стеке. Правда, для этого пришлось использовать три

команды вместо одной. Но и устранение недостатка команды FX^y любым другим способом (использование адресуемых регистров, команд принудительного перемещения чисел в стеке) также потребует дополнительно не менее двух команд.

8.4.2. ИЗМЕНЕНИЕ ЕСТЕСТВЕННОГО ПОРЯДКА ПЕРЕМЕЩЕНИЯ ЧИСЕЛ В СТЕКЕ

Как известно, для выполнения одноместной операции число должно находиться в регистре X, а двухместной — числа должны находиться в регистрах X и Y. В тех случаях, когда необходимо выполнить одноместную или двухместную операцию, а хотя бы один из operandов не находится в указанных регистрах (но находится в других регистрах стека) приходится прибегать к принудительному перемещению чисел в стеке в нужном направлении, вопреки естественному порядку. Для этой цели в ПМК предусмотрены специальные команды перемещения и перестановки чисел в регистрах стека, выполнение которых предписывается нажатием соответствующих клавиш, перечень и назначение которых приводится ниже (табл. 3.4).

Таблица 3.4

Команды перемещения чисел в стеке

Назначение команды	Обозначение команды клавиши	Перемещение чисел в стеке в результате выполнения команды
Разделение исходных данных, вводимых с пульта ПМК, и/или перемещение чисел в стеке от X к T	B↑ или B!	
Вращение стека, кольцевое перемещение чисел в стеке в обратном направлении.	F ↘ или F.	
Перестановка чисел в операционных регистрах X и Y	↔	

Приложение. Условимся в дальнейшем пользоваться упрощенным обозначением клавиши F ↘, обозначая ее как F. Последнее обозначение не вызывает трудностей, поскольку на самой клавише в качестве основного ее обозначения помещена точка.

Пусть начальное состояние стека будет (1,2,3,4). Посмотрим теперь, как оно будет изменяться после выполнения каждой из команд перемещения чисел в стеке.

Команда B!. Начальное состояние стека — (1,2,3,4):

00. B! (1, 1, 2, 3)
01. B! (1, 1, 1, 2)
02. B! (1, 1, 1, 1)

Для заполнения всех регистров стека содержимым регистра X достаточно три раза нажать на клавишу B!.

Примечание. Команду B! рекомендуется использовать только в режиме ручных вычислений и при вводе чисел. В программируемом режиме лучше всего избегать пользоваться этой командой. Вместо нее для разделения чисел более целесообразно использовать команду КНОП (нет операции), а для принудительного перемещения чисел в стеке — команды ↔ и F..

Команда F. Начальное состояние стека — (1,2,3,4):

00. F. (2, 3, 4, 1)
01. F. (3, 4, 1, 2)
02. F. (4, 1, 2, 3)
03. F. (1, 2, 3, 4)

Полный оборот стека достигается путем четырехкратного нажатия клавиши F.

Команда ↔. Начальное состояние стека — (1,2,3,4):

00. ↔ (2, 1, 3, 4)
01. ↔ (1, 2, 3, 4)

При двукратном нажатии на клавишу ↔ полностью восстанавливается содержимое регистров X и Y, т. е. становится таким, каким оно было до нажатия. Это часто используется, когда необходимо проконтролировать содержимое регистра Y.

Используя команды перемещения чисел в стеке ↔ и F., можно очистить его от «заедающего патрона» — значения показателя степени y при выполнении операции возведения в степень FX^y . Обозначим $N = M^p$. Для вычисления необходимо выполнить следующие действия:

- ВВЕСТИ $p!$ M (M, p)
00. $FX^y (M^p, p)$

Дальнейшему использованию стековой памяти, как показано выше, мешает наличие в стеке показателя степени, оставшегося после выполнения операции в регистре Y стека. Для того чтобы его удалить, необходимо после команды возведения в степень выполнить две следующие команды:

01. ↔ (p, M^p)
02. F. (M^p)

Покажем, как будет изменяться состояние стека при многочленном выполнении операции возведения в степень и с удалением из стека значения показателя степени после выполнения

каждой операции. Пусть начальное состояние стека будет (4,3,2,1). Тогда

- | | |
|--|--|
| 00. FX ^y (64.000004, 3, 2, 1) | 05. F. (4095.9999, 1, 3, 2) |
| 01. ↔ (3,64.000004, 2, 1) | 06. FX ^y (4095.9999, 1, 3, 2) |
| 02. F. (64.000004, 2, 1, 3) | 07. ↔ (1.4095.9999, 3, 2) |
| 03. FX ^y (4095.9999, 2, 1, 3) | 08. F. (4095.9999, 3, 2, 1) |
| 04. ↔ (2,4095.9999, 1, 3) | |

В аналогичном предыдущем примере многократно выполнялось возвведение в одну и ту же третью степень. Перемещение стека при этом не происходило. В только что рассмотренном примере благодаря удалению старого показателя степени после каждой операции каждый раз производится возвведение в новую степень. В этом случае состояние стека изменяется точно так, как и при обычной двухместной операции (+ — × :).

Упражнение

3.4.1. Задано начальное состояние стека (1, 2, 3, 4). Используя команды перемещения чисел в стеке, получить все возможные комбинации расположения чисел в регистрах X и Y.

3.5. ПРИМЕРЫ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ СТЕКОВОЙ ПАМЯТИ

Вычисления с использованием конкретных чисел и применением только двух регистров стековой памяти — операционных регистров X и Y рассматривались в п. 2.5. Здесь будут приведены примеры, предусматривающие действия над значениями переменных. При этом будут использованы более чем два регистра стековой памяти. Условимся в приведенных ниже примерах все промежуточные результаты записывать только в регистры стековой памяти.

Примеры:

I. Вычисление площади круга $S = \pi D^2/4$ с использованием стековой памяти.

- | |
|---------------------------------------|
| ВВЕСТИ D (D) |
| 00. FX ^x (D ²) |
| 01. Fπ (π, D ²) |
| 02. × (4, π, D ²) |
| 03. + (π4, D ²) |
| 04. × (πD ² /4) |

ВЫДАТЬ S

2. Вычисление площади треугольника по формуле Герона с использованием адресуемой и стековой памяти.

В п. 3.2 (пример 2) была составлена программа вычисления площади треугольника по формуле Герона с использованием только адресуемой памяти ПМК. Покажем, насколько упростится программа вычислений, если для хранения промежуточных результатов использовать стековую память. Пусть необходимо вычислить

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

Распределение памяти:

$$R0 = a; R1 = b; R2 = c; R3 = p; RX = S.$$

Программа вычислений:
 ВВЕСТИ R0 = a, R1 = b, R2 = c
 00. пX0 (a)
 01. пX1 (b, a)
 02. пX2 (c, b, a)
 03. + (b + c, a)
 04. + (a + b + c)
 05. 2 (2, a + b + c)
 06. ÷ ((a + b + c)/2)
 07. хП3 (R3 = p)
 08. пX3 (p, p)
 09. пX0 (a, p, p)
 10. — (p — a, p)

11. × (p (p — a))
 12. пX3 (p, p (p — a))
 13. пX1 (b, p, p (p — a))
 14. — (p — b, p (p — a))
 15. × (p (p — a) (p — b))
 16. пX3 (p, p (p — a) (p — b))
 17. пX2 (c, p, p (p — a) (p — b))
 18. — (p — c, p) (p — a) (p — b))
 19. × (p (p — a) (p — b) (p — c))
 20. F V ($\sqrt{p(p-a)(p-b)(p-c)}$)
 ВЫДАТЬ S

По предлагаемой программе выполним расчет площадей трех треугольников Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a+b+c)/2$. Результаты расчета сведем в табл. 3.5.

Результаты расчета, приведенные в табл. 3.4 при одинаковых значениях сторон треугольников a , b , c , полностью совпадают с результатами расчета из табл. 3.1 и 3.2, что подтверждает правильность первых. Сравним примеры расчета площадей треугольников с использованием только адресуемой памяти (пример 2в п. 3.2) и с использованием адресуемой и стековой памяти (пример 2в п. 3.5). В первом случае пришлось использовать 7 регистров адресуемой памяти, а во втором — только 4. В первом случае программа расчета состояла из 26 команд, а во втором — только из 20. Таким образом, использование стековой памяти значительно сокращает как длину программы, так и объем памяти, необходимой для производства вычислений.

Таблица 3.5

Расчет площади треугольника с использованием адресуемой и стековой памяти

a , м	b , м	c , м	S , м ²
0.25	0.50	0.60	6.1361505—02
0.50	1.00	1.20	2.4544602—01
1.00	2.00	2.5	9.4991775—01

Упражнения

3.5.1. Вычислить следующие выражения с использованием стековой памяти для хранения промежуточных результатов. Сравнить полученные программы вычислений и результаты с таковыми в упр. 3.2.1:

$$1) T = \frac{24.3 \cdot 2.6 + 16.8 \cdot 3.3}{0.22 \cdot 5.65 + 8.58 \cdot 0.073} =$$

$$2) M = 20^5 + \sqrt[3]{14964} - 16^6 =$$

Примечание. $\sqrt[3]{\dots}$ вычисляется как (...) 1/3.

$$3) T = \frac{343.458 + 507.985 + 387.174}{114 + 234 + 859 + 484}.$$

3.4.2. Дано:

$$P = \sin^4 \varphi + \cos^4 \varphi - (1 - \operatorname{tg}^2 \varphi) / \operatorname{tg}^2 \varphi.$$

Требуется:

- Составить программу для вычисления P при любых φ .
- Вычислить P для $\varphi = 72^\circ$ (сравнить с упр. 3.2.2).

3.5.3. Составить программу вычисления выражения с использованием стековой памяти ПМК (сравнить с упр. 3.2.3)

$$P = \left(\lg \frac{(1/\sqrt{3} + 3\sqrt{5} - \operatorname{arctg} 1.2) \cdot 10^{-3}}{\ln 2 \cdot \ln 3 + \ln 4 \cdot \ln 5} - 2.2^9 \right)^2 + 2.2^9.$$

(Значение arctg дается в радианах).

3.6. РЕКОМЕНДАЦИИ ПО ИСПОЛЬЗОВАНИЮ АДРЕСУЕМОЙ И СТЕКОВОЙ ПАМЯТИ ПМК

Использование адресуемой и стековой памяти ПМК во многих случаях избавляет от записи на бумаге промежуточных результатов и ручного повторного ввода их в ПМК, что существенно ускоряет процесс вычислений и создает предпосылки к его автоматизации. Стековая память ПМК обладает двумя основными достоинствами: она обеспечивает автоматическое запоминание результатов предыдущих действий и использование их для выполнения последующих, а также уменьшает время, затрачиваемое на обращение к памяти (отсутствуют команды обращения к памяти и, кроме того, время выборки числа из стекового регистра значительно меньше, чем из адресуемого регистра оперативной памяти).

Использование стековой памяти требует, чтобы до и после выполнения каждой команды было известно точное состояние всех стековых регистров. Однако точный учет содержимого всех четырех стековых регистров в процессе решения задачи затруднителен, так как после выполнения любой команды на экране индикатора отображается лишь содержимое одного регистра X . Это вызывает необходимость заранее составлять программу решения задачи, в которой определить содержимое всех регистров стека практически можно в автоматическом режиме использования ПМК, когда программа заранее составляется и обдумывается. При расчетах, выполняемых вручную, использование стековой памяти наиболее эффективно для простейших задач, в которых достаточно следить за состоянием первых двух регистров памяти, а также для заранее отработанных типовых задач, алгоритмы решения которых будут рассмотрены ниже.

Поэтому в ручном режиме стековую память целесообразно применять при заранее составленной программе вычислений, в которой после каждого шага обязательно должно быть зафиксировано состояние регистров стека. В автоматическом режиме, когда программа уже введена в оперативную память ПМК, использование стековой памяти трудностей не вызывает, поэтому нет причин, чтобы не использовать ее возможности. Наиболее целесообразным и эффективным является комбинированное использование оперативной и стековой памяти. При этом входной поток чисел будет поступать в стек из оперативной памяти, а поток промежуточных

результатов из регистров Z и T стека — в операционные регистры X и Y . В этом случае количество команд, необходимых для решения задачи, будет наименьшим, а скорость вычислений — наибольшей, что особенно важно при работе ПМК в автоматическом (программируемом) режиме.

Глава 4. ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ С МНОГОКРАТНО ПОВТОРЯЮЩИМИСЯ ОПЕРАЦИЯМИ

В предыдущей главе было показано, как с помощью ПМК выполнять вычисления, характеризующиеся линейной последовательностью операций. Такие вычисления описываются программами, которые выполняются в порядке возрастания номеров команд. Вычисления такого типа называются линейными и являются наиболее простыми. Однако на практике задач с линейной схемой вычислений очень мало (встречаются в основном линейные участки вычислений других типов). Чаще всего приходиться иметь дело с повторениями, когда многократно выполняется некоторая последовательность операций над различными значениями одних и тех же независимых переменных (например, расчеты различных строк одной и той же таблицы).

Очень многие задачи содержат также ветвления, когда выполняется тот или иной участок вычислительного процесса в зависимости от соблюдения или несоблюдения некоторого условия.

Наиболее эффективно и целесообразно применять ПМК для решения таких задач, в которых последовательности элементарных вычислительных операций повторяются с различными значениями исходных и промежуточных данных. Например:

задачи, решаемые итерационными методами;
расчеты по рекуррентным формулам;
расчеты таблиц, каждая строка которых вычисляется по одному и тому же правилу (алгоритму);
относительно простые задачи, решаемые методом статистических испытаний (простые статистические модели).

Эти задачи являются, как правило, более сложными и серьезными, чем ранее рассмотренные, где вычисления были линейными. Но чтобы решить сколько-нибудь серьезную задачу, надо прежде всего продумать действия, позволяющие получить искомый результат, приспособить их к возможностям и особенностям ПМК и только после этого переходить к написанию программ вычислений в виде последовательности команд. Однако запись программ в виде последовательности команд хороша, чтобы не ошибиться в нажатии клавиш, но она мало пригодна для того, чтобы сообщить другому пользователю или вспомнить самому, что же решает эта программа. Между тем понимание программы необходимо всегда, хотя бы для того, чтобы знать, как исправить программу при обнаружении ошибок или откорректировать ее при изменении

условий задачи. Все это приводит к необходимости для любой сколько-нибудь серьезной задачи предварительного составления проекта (плана), ориентированного на понимание его пользователем. Такой проект (план) программы обычно называется *алгоритмом решения задачи*.

До последнего времени алгоритмы решения задач было принято представлять в графической форме в виде блок-схем. Однако более точно и более строго любой алгоритм можно записать с помощью алгоритмического языка (языка описания алгоритмов). Точность и строгость алгоритмического языка позволяет более успешно использовать разработанные алгоритмы для дальнейшего составления программ по сравнению с их записью на языке блок-схем. В нашей стране наиболее распространенным языком описания алгоритмов является алгоритмический язык, предложенный академиком А. П. Ершовым и изучаемый в рамках нового предмета общеобразовательной школы «Основы информатики и вычислительной техники» [14]. Этот язык очень удобен для описания вычислений на ПМК в ручном режиме, поэтому в предлагаемой главе будем ориентироваться на него с необходимыми дополнениями и изменениями, обусловленными использованием ПМК в ручном режиме работы. С этой целью в дальнейшем будем широко использовать конструкции указанного алгоритмического языка и примеры их применения, приведенные в [14]. Для краткости этот язык будем называть также лексиконом, подчеркивая тем самым его общее назначение для первоначального описания исходных алгоритмов пользователя.

4.1. АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ. ЯЗЫК ОПИСАНИЯ АЛГОРИТМОВ

Под алгоритмом решения задачи понимают правило, сформулированное на некотором языке и определяющее процесс переработки допустимых исходных данных в искомый результат. Описание любого алгоритма представляет собой некоторую последовательность предложений, каждое из которых выражает строго определенное законченное действие, предписываемое к исполнению. Цепочка таких действий, ведущих от исходных данных к искомому результату, называется алгоритмическим процессом, а каждое действие — его шагом или командой.

Здесь важно отметить, что алгоритм предполагает наличие его исполнителя и включает в себя указания, что он должен сделать, чтобы, имея исходные данные, получить искомый результат. Отсюда вытекает, что алгоритм должен быть понятен исполнителю, иначе он не может быть выполнен. В случае использования ПМК исполнителем является как человек — пользователь ПМК, так и программируемый микрокалькулятор. Поэтому алгоритм должен быть ориентирован как на понимание его человеком, так и на восприятие его (в той части, в которой осуществляется автоматизация вычислений) программируемым микрокалькулятором. Алгоритми-

ческий язык, предложенный академиком А. П. Ершовым (лексикон) [14], хотя и ориентирован в первую очередь на человека, но имеет своей конечной целью последующее решение задачи на персональной ЭВМ с использованием языка программирования типа РАПИРА.

Поскольку мы ориентируемся на использование ПМК, а не персональной ЭВМ, то это обстоятельство потребует внесения соответствующих изменений и дополнений в конструкции алгоритмического языка (лексикона). Эти изменения и дополнения заключаются в следующем:

1. В ПМК производятся действия только над величинами вещественного типа. Форму представления чисел, выдаваемых на экран индикатора (целые, вещественные, в естественной или показательной форме), пользователь может изменять. Поэтому указания типов переменных всюду опускаются. Исключается также список аргументов и результатов с указанием типа, помещаемый в скобках после названия алгоритма.

2. Перечни аргументов, результатов и списки промежуточных величин одновременно используются для распределения памяти ПМК. Каждому аргументу, результату и промежуточной величине ставится в соответствие регистр памяти ПМК, где должны храниться их значения:

регистр = имя,

где регистр — это имя регистра, состоящее из буквы R и его номера (внутренняя переменная ПМК), а имя — имя переменной, ее обозначение в соответствующей формуле (внешняя переменная для ПМК). Например, R1 = m, R2 = n.

Необходимость включения распределения памяти в алгоритм обусловлена тем, что в ПМК по сравнению с микроЭВМ емкость памяти сильно ограничена (по крайней мере, на два порядка меньше, чем в микроЭВМ). И вопрос о том, хватит ли имеющейся памяти, равносителен вопросу, можно ли решать задачу с помощью ПМК, а следовательно, нужно ли составлять алгоритм ее решения. Поэтому распределение памяти должно быть выполнено в самом начале алгоритма решения задачи.

3. С целью упрощения написания в качестве знака присваивания всюду употребляется знак равенства =, за исключением случая его использования в условных выражениях. В том случае, когда правая и левая части условия соединены знаком =, последний воспринимается как знак равенства, а не как знак присваивания.

4. Дополнительно используются команды ввода чисел с пульта ПМК ВВЕСТИ (сокращенно ВВ) и выдачи результатов, отображаемых на экране индикатора ВЫДАТЬ (сокращенно ВЫ).

5. Дополнительно вводится команда НАЖАТЬ (сокращенно НЖ), предписывающая нажатие тех или иных клавиш на пульте управления.

Алгоритмический язык с перечисленными изменениями и дополнениями можно рассматривать как некоторую версию алгоритмического языка (лексикона), ориентированную на последующее использование ПМК. Рассмотрим основные конструкции указанной версии.

4.1.1. ОСНОВНЫЕ КОНСТРУКЦИИ АЛГОРИТМИЧЕСКОГО ЯЗЫКА

Общий вид алгоритма:

```

алг название
арг перечень аргументов
рез перечень результатов
нач список промежуточных величин
| серия
кон [14]
алг КВУР
арг RA = a, RB = b, RC = c
рез RX = xl, RX = x2, RX = ЕГГОГ
нач RD = D
ВВЕСТИ RA = a, RB = b, RC = c
D = b2 - 4 * a * c
если D < 0
    то ВЫДАТЬ ЕГГОГ
    иначе x1 = (-b + D)/(2 * a)
        x2 = (-b - D)/(2 * a)
все
кон

```

Служебные слова, имеющие строго определенное значение во всех конструкциях алгоритмического языка, в алгоритме выделяются подчеркиванием и/или жирным шрифтом. К ним относятся: алг, арг, рез, нач, кон, если, то, иначе, все и др. Сюда же следует отнести слова ВВЕСТИ, ВЫДАТЬ, НАЖАТЬ, используемые в командах-предписаниях, выполняемых пользователем ПМК.

Алгоритм состоит из двух частей: заголовка, включающего строки со служебными словами алг, арг, рез, и собственно алгоритма, начинающегося со слова нач и заканчивающегося словом кон (конец). В заголовке алгоритма решения квадратного уравнения указывается, что этот алгоритм называется КВУР (квадратное уравнение), аргументами его являются значения переменных a , b , c , для которых выделяются регистры памяти ПМК RA, RB, RC соответственно, а результатами — корни квадратного уравнения x_1 и x_2 и признак ЕГГОГ, указывающий на отсутствие решения (корни мнимые). Для результатов выделены регистры стековой памяти RX и RY, причем один и тот же регистр X выделяется и для корня x [14] и для признака ЕГГОГ, поскольку эти значения используются в разных ветвях алгоритма.

После служебного слова нач указывается, что в алгоритме используется промежуточная величина D , для хранения которой выделяется регистр RD. После этого следует серия команд, ко-

торые должен выполнить исполнитель алгоритма (символ * в командах означает умножение). Команды алгоритма записываются, как правило, на одной строке. Переход на другую строку означает начало другой команды. Если же на одной строке записывается более одной команды, то они отделяются друг от друга в этой строке точкой с запятой. Продолжение команды на другой строке записывается правее (как минимум, на три позиции от ее начала на предыдущей строке).

Команда установления единицы измерения аргументов тригонометрических функций:

{РАДИАНЫ}
{ГРАДЫ}
{ГРАДУСЫ}

Здесь и в дальнейшем {...} означает, что должна быть выбрана одна из помещенных в фигурные скобки конструкций.

Команда определяет единицу измерения аргумента тригонометрических функций, вводимого с пульта ПМК в регистр X. Она предписывает выполнить установку переключателя Р—ГРД—Г в соответствующее положение. Команда обычно записывается сразу же после заголовка алгоритма и сохраняет свое действие вплоть до появления в алгоритме другой команды РАДИАНЫ—ГРАДЫ—ГРАДУСЫ, изменяющей ранее установленную единицу измерения.

Команда ввода исходных данных с пульта ПМК:

{ВВЕСТИ} список вводимых величин [(стек)]

Здесь [...] означает, что помещенная в квадратные скобки конструкция может быть опущена.

Примеры:

- 1) ВВЕСТИ RA = a, RB = b, RC = c;
- 2) ВВ RD = n, R1 = m, RX = q;
- 3) ВВ D 1 λ (λ, D).

Команда предписывает набор на клавиатуре числа или последовательности чисел и запись их в регистры, прямо или косвенно указаные в команде. В первом и втором примерах регистры указаны явно: $a \rightarrow RA$, $b \rightarrow RB$, $c \rightarrow RC$, а в третьем примере — неявно. Здесь сначала вводится значение D , затем признак окончания ввода B!, а после — значение λ . В результате ввода значение λ будет помещено в регистр X, а D — в регистр RY, что обозначается состоянием стека, помещенным в круглые скобки.

Команда выдачи результатов:

{ВЫДАТЬ} список результатов

Примеры:

- 1) ВЫДАТЬ RX = S, R0 = a, R1 = b;
- 2) ВЫ S, K, M, P.

Команда предписывает пользователю прочитать на экране индикатора и записать их на бумаге в случае необходимости. В первом примере в списке результатов явно указаны регистры, из которых следует прочитать результаты. В частности, первый элемент списка указывает, что значение S следует прочитать в регистре X , содержимое которого непосредственно отображается на экране индикатора, а значение a прочитать в регистре $R0$. b — в регистре $R1$. Во втором примере регистры, в которых хранятся результаты, указаны неявно. Значения результатов считываются при этом из регистров стековой памяти, а именно: S — из RX , K — из RY , M — из RZ , p — из RT .

Команда присваивания:

имя = выражение.

где имя — это имя (наименование) некоторой переменной.

В результате выполнения команды присваивания значение этой переменной становится равным указанному в правой части значению выражения. При записи выражений могут быть использованы числовые константы, имена переменных, скобки, знаки выполнения операций $+$ — $/$ * (умножить) ** (возвести в степень), а также стандартные функции, вычисление которых предусмотрено в ПМК. Например, $\sin(x)$, $\cos(x)$, $\tg(x)$, $\arcsin(x)$, $\arccos(x)$, $\arctg(x)$, $\text{sqr}(x)$ (квадратный корень), $\lg(x)$, $\ln(x)$, $\exp(x)$ (e^x) и др. Как видим, аргумент функций помещается в скобках. Полный перечень функций, используемых в ПМК, дан в прил. 1.

В выражениях допускается также использование общепринятых математических обозначений функций и переменных, а также знаков математических операций над ними. В частном случае выражения могут состоять из одного имени переменной или одного числа. В качестве аргументов стандартных функций могут выступать не только переменные величины, но и выражения.

Команда присваивания выполняется в следующей последовательности:

— вычисляется численное значение выражения, стоящее в правой части;

— полученное значение (число) присваивается переменной, стоящей в левой части, а предыдущее ее значение стирается.

Правая часть выражения называется источником присваивания, а левая часть — ее получателем. В источник в качестве его элементов могут входить только те переменные, значения которых были определены (введены или вычислены) ранее в алгоритме предыдущими предложениями и операторами. Элементами источника могут быть только те операции, которые могут быть выполнены на ПМК в автоматическом режиме.

Как уже отмечалось (п. 3.1), применительно к ПМК процесс выполнения операции присваивания заключается в записи значе-

ния правой части (выражения) в регистр памяти, выделенный для хранения переменной, стоящей в левой части.

П р и м е ч а н и е. Следует еще раз подчеркнуть, что знак $=$ не является в команде присваивания знаком равенства. Это знак присваивания. До выполнения операции присваивания значение переменной в левой части не равно значению выражения. Оно лишь приравнивается значению переменной, т. е. левая и правая части команды присваивания становятся равными лишь после вычисления выражения и выполнения операции присваивания.

П р и м е р ы:

$$1) x = \frac{-b + \sqrt{D}}{2 * a}$$

$$2) x = (-b + \sqrt{D}) / (2 * a)$$

Вторая запись является более предпочтительной, так как она располагается в одну строку. (Запись $x_2 = (-b + \sqrt{D}) / 2 * a$ была бы неправильной, так как при этом предусматривался бы другой порядок вычисления выражения, а именно: $x_2 = ((-b + \sqrt{D}) / 2) * a$, а это не одно и то же).

$$3) x = (-b + \text{sqr}(D)) / (2 * a).$$

Здесь D — заменяется функцией sqr (сокращение английских слов *square root* — квадратный корень), вычисляемой с помощью команды ПМК FV .

Команда НАЖАТЬ:

{НАЖАТЬ} список команд ПМК [(стек)]

НАЖАТЬ	$\text{НЖ}_\text{Fx}^2 \text{ F}\pi \text{ } 4 \text{ } \div \text{ } \times$
00. $\text{Fx}^2(D^2)$	$\text{НЖ}_\text{00. Fx}^2 \text{ 01. F}\pi \text{ } 02. 4 \text{ } 03. \div \text{ } 04. \times$
01. $\text{F}\pi(\pi, D^2)$	
02. $4(4, \pi, D^2)$	
03. $\div(\pi/4, D^2)$	
04. $\times(\pi D^3/4)$	

П р и м е ч а н и я:

1. Элементы списка команд могут содержать номер команды, записываемый всегда двухзначным числом. После номера команды всегда ставится точка.

2. После записи каждой команды вслед за ней может оказываться состояние регистров стека, которое будет иметь место после ее выполнения.

3. Команды ПМК после слова НАЖАТЬ могут записываться как вдоль строки слева направо, так и столбцом сверху вниз.

4. Команды ПМК могут включать в себя и числовые константы. При этом каждой командой должен предписываться ввод только одного символа: цифры, точки или знака $/$ — $/$. Вся константа вводится последовательностью команд.

Пример линейного алгоритма. Алгоритм называется линейным, если его команды выполняются в том же порядке, в котором они записаны. Последовательность операций, определяемая таким алгоритмом, называется линейным вычислительным процессом или *следованием*. Рассмотрим алгоритм такого процесса на примере

вычисления площади треугольника по формуле Герона. Он будет выглядеть следующим образом:

АЛГОРИТМ 4.0

```
алг ГЕРОН-4.0 *  $S = \sqrt{p(p-a)(p-b)(p-c)}$ , где  $p = (a+b+c)/2$  *
    арг R0 = a, R1 = b, R2 = c
    рез RX = S
нач R3 = p
    ВВЕСТИ R0 = a, R1 = b, R2 = c
     $p = (a+b+c)/2$ 
```

НАЖАТЬ
00. пX0(a)
01. пX1(b, a)
02. + (a + b)
03. пX2(c, a + b)
04. + (a + b + c)
05. 2 (2, a + b + c)
06. ÷ ((a + b + c)/2)
07. хП3 (R3 = p)

$S = \sqrt{p * (p - a) * (p - b) * (p - c)}$
НАЖАТЬ
08. пX3(p, p)
09. пX0(a, p, p)
10. — (p - a, p)
11. × (p (p - a))
12. пX3(p, p (p - a))
13. пX1(b, p, p (p - a))
14. — (p - b, p (p - a))
15. × (p (p - a) (p - b))
16. пX3(p, p (p - a) (p - b))
17. пX2(c, p, p (p - a) sp - b))
18. — (p - c, p (p - a) (p - b))
19. × (p (p - a) (p - b) (p - c))
20. F V ($\sqrt{p (p - a) (p - b) (p - c)}$)

ВЫДАТЬ S

кон

В левой части листа помещен алгоритм решения задачи, т. е. план ее решения, предназначенный для пользователя, но не воспринимаемый непосредственно ПМК, а в правой половине дана детализация алгоритма — программа решения задачи на ПМК. Она представлена в виде последовательности команд ПМК. При этом каждая команда включает в себя номер, обозначение и состояние стека после ее выполнения.

Каждой команде присваивания алгоритма соответствует команда НАЖАТЬ, предписывающая пользователю выполнить соответствующий набор команд на клавиатуре пульта управления ПМК.

Команда ветвления. Предложения алгоритмического языка команды могут быть не только простыми, но и сложными. Ветвление — это составная команда, которая в зависимости от соблюдения или несоблюдения некоторого заданного условия позволяет выбрать для выполнения ту или иную серию команд:

```
если условие
|   то серия 1
|   иначе серия
все [14]
```

```
если a ≥ b
|   то max = a
|   иначе max = b
все
```

Простое условие имеет следующий вид:

выражение внак выражение
 $a + b > c + d$

Составное условие состоит из простых условий, соединяемых служебными словами и/или:

$x \leq -1$ или $x \geq 1$;
 $x > -1$ и $x \leq 2$.

Выражения в условиях составляются и вычисляются по тем же правилам, что и в командах присваивания. В качестве знаков условия используются $\geq (> =)$, $>$, $<$, $\leq (< =)$, $=$, $\neq (/ =)$. Знак $=$ используется в условиях только как знак равенства, а не знак присваивания.

Помимо ветвлений условия также используются в командах выбора и повторения.

Ветвление выполняется следующим образом:

проверяется условие;

если оно соблюдается, то выполняется серия 1 (ветвь то);

если оно не соблюдается, то выполняется серия 2 (ветвь иначе).

Выполнение ветвления заканчивается после того, как будет выполнена серия команд, входящая в одну и только одну из ветвей (серия 1 или серия 2). Точкой входа ветвления является служебное слово если, а точкой выхода — служебное слово все.

В приведенном примере команды ветвления при $a \geq b$ значением переменной тах становится число a . В противном случае (при $a < b$) значением максимальной величины становится число b .

Ветвь иначе может быть опущена. Тогда в случае соблюдения условия выполняется ветвь то, а в случае его несоблюдения — производится выход из ветвления в точку все. Если в ветвлении ветвь иначе исключена, то мы имеем дело с сокращенной формой ветвления:

```
если условие
|   то серия
все
```

если $x < 0$
| то $x = -x$
все

Рассмотрим теперь пример алгоритма с ветвлением. Пусть необходимо вычислить $\varphi = \operatorname{arctg}(\cos \psi / (1 - \operatorname{tg} \psi))$. При $1 - \operatorname{tg} \psi = 0$ вычислить φ невозможно, так как в этом случае знаменатель дроби равен нулю, а аргумент арктангенса равен бесконечности. Однако арктангенс от бесконечности равен $\pi/2$. Следовательно, при $1 - \operatorname{tg} \psi = 0$ следует не вычислять значение аргумента, а сразу же присвоить искомой функции значение $\pi/2$. Это можно записать в виде алгоритма:

алг ФАЗА
арг R1 = ψ
рез RX = φ

нач
если $1 - \operatorname{tg} \psi \neq 0$
то $\varphi = \arctg(\cos \psi (1 - \operatorname{tg} \psi))$

НАЖАТЬ
00. пXI (ψ)
01. Fcos ($\cos \psi$)
02. 1 (1, $\cos \psi$)
03. пXI (ψ , 1, $\cos \psi$)
04. Ftg ($\operatorname{tg} \psi$, 1, $\cos \psi$)
05. — ($1 - \operatorname{tg} \psi$, $\cos \psi$)
06. ÷ ($\cos \psi (1 - \operatorname{tg} \psi)$)
07. Ftg⁻¹ (ψ)

иначе ВЫДАТЬ $\pi/2$

все

кон

В рассмотренном алгоритме детализируются и преобразуются в команды ПМК только команды присваивания алгоритма. Выполнение ветвления осуществляется пользователем ПМК вручную.

Команда выбора. Ветвление — распределение вычислений только на две ветви. Однако вычислительный процесс часто требуется распределить более чем на две ветви. В таком случае используется команда выбора:

выбор
при условие 1: серия 1
при условие 2: серия 2
при условие 3: серия 3
...
при условие n : серия n
иначе серия
все [14]

выбор
при $x \geq 0.5$: $y = \sqrt{x}$
при $x > 0$ и $x < 0.5$: $y = \cos x$
при $x < 0$: $y = \sin x$
иначе $y = 1$
все

Выполнение команды выбора осуществляется следующим образом. Проверяется *условие 1*. Если оно соблюдается, то выполняются команды, входящие в *серию 1*, и на этом выполнение команды выбора заканчивается. Происходит выход в точке *все*. Если *условие 1* не соблюдается, то проверяется *условие 2*. Если последнее соблюдается, то выполняются команды *серии 2* и происходит выход из команды выбора. Если же и *условие 2* не соблюдается, то проверяется *условие 3* и т. д. Если же ни одно из условий не соблюдается, то выполняется *серия* команд, входящая в ветвь *иначе*, после чего происходит выход из команды.

Возможна также сокращенная форма команды выбора без ветви *иначе* (она более удобна для алгоритма, ориентированного на ПМК). В последнем случае, если ни одно из условий не соблюдается, происходит выход из команды выбора и переход к следующей команде алгоритма.

В приведенном примере условие 2 является сложным, состоящим из двух условий: $x > 0$ и $x \leq 0.5$. Соблюдение этого условия будет заключаться в одновременном выполнении обоих условий. Если хотя бы одно из них не соблюдается, то не соблюдается в целом и сложное условие.

Команды повторения. Повторение, или цикл, — это составная команда, предписывающая многократное выполнение некоторой серии команд. Различают три типа команд повторения:

пока;
до;
с параметром.

1. Команда повторения типа **пока**:

пока	условие	пока	$x \geq 10$
иц		иц	
	серия		$x = x - 10$
кц			

Здесь *иц* и *кц* обозначают соответственно начало и конец одного цикла.

При выполнении команды повторения типа **пока** входящая в нее *серия* команд выполняется столько раз, сколько нужно, чтобы *условие* перестало соблюдаться. Если *условие* не соблюдается с самого начала, то *серия* не будет выполнена ни одного раза. В нашем примере значение x будет уменьшаться каждый раз на 10, пока оно не станет равным 10. Если начальное значение переменной — целое положительное число, то после выполнения команды повторения ее новым значением станет остаток от деления исходного числа на 10 [14].

2. Команда повторения типа **до**:

иц		иц	
	серия		$S = S + v$
кц		$v = v/x$	
до	условие	кц	
		до	$v < 10^{-3}$

В отличие от команды повторения предыдущего типа здесь *серия* команд, входящая в команду повторения, всегда выполняется хотя бы один раз. С каждым повторением значение v уменьшается в x раз. В конце каждого повторения проверяется его *условие*. Если оно не соблюдается, то происходит возврат к началу и процесс продолжается. Как только условие станет соблюдаться, то выполнение команды повторения заканчивается и происходит переход к следующей команде алгоритма.

Из-за особенностей построения ПМК (или, как говорят применительно к ЭВМ, архитектуры) команда **до** более предпочтительна, чем команда **пока**.

Команда повторения с параметром. Величину, изменяющуюся от повторения к повторению, но сохраняющую свое значение в пределах одного цикла, принято называть его параметром. Серию команд, выполняемых в каждом повторении, часто называют его телом. При этом параметр изменяется на постоянную величину, называемую шагом. В последнем случае удобно использовать команду повторения с параметром

```
для x от Xнач до Xкон шаг Xшаг
иц
| серия
кц
```

Здесь x — параметр (переменная величина, дискретно изменяющаяся от повторения к повторению); $X_{\text{нач}}$, $X_{\text{кон}}$ — выражения, определяющие начальное и конечное значение параметра; $X_{\text{шаг}}$ — выражение, определяющее шаг изменения параметра (если шаг изменения равен единице, то служебное слово шаг вместе со значением шага $X_{\text{шаг}}$ может быть опущено).

Выполнение команды повторения с параметром происходит следующим образом:

- вычисляются значения выражений $X_{\text{нач}}$ и $X_{\text{кон}}$;
- переменной x последовательно присваиваются значения x , $x + X_{\text{шаг}}$, $x + 2X_{\text{шаг}}$, ..., $x + nX_{\text{шаг}}$ и для каждого из этих значений выполняется серия команд, заключенных между иц и кц;

— как только x и $x_{\text{шаг}}$ превзойдет значение $X_{\text{кон}}$, серия команд между иц и кц не выполняется, а происходит выход из команды повторения — переход к выполнению следующей команды алгоритма.

Если $X_{\text{нач}} = X_{\text{кон}}$, то команда повторения выполняется только один раз. Если $X_{\text{нач}} < X_{\text{кон}}$, то команда присваивания может быть выполнена только при отрицательном значении $X_{\text{шаг}}$.

Пример:
для x от 0,1 до 6.28 шаг 0.01

```
иц
| ВВЕСТИ x
| y = sin x
| ВЫДАТЬ y
кц
```

для D от 4 до 10

```
иц
| ВВЕСТИ R1 = D
| G{ДБ} = 10 * lg (K (πD/λ) 2)
| ВЫДАТЬ G{ДБ}
кц
```

Сложное повторение (сложный цикл). В рассмотренных ранее примерах мы имели дело с простыми повторениями, в которых был только один параметр цикла. Существуют также сложные повторения, с несколькими параметрами. Они представляют собой несколько вложенных друг в друга повторений.

Пусть, например, требуется определить площадь, прямоугольника для трех значений каждой из сторон: $a = 1, 2, 3$ м и $b = 1, 2, 3$ м. Тогда предложение, предписывающее выполнение таких вычислений, будет выглядеть следующим образом:

```
для a от 1 до 3
иц
| ВВЕСТИ R1 = a
| для b от 1 до 3
| иц
| | ВВЕСТИ R2 = b
| | S = a * b
| | ВЫДАТЬ S
| кц
кц
```

В случае сложного повторения (цикл в цикле) сначала фиксируются значения параметра внешнего повторения (в данном случае a). Затем для каждого фиксированного значения параметра внешнего повторения вычисляется тело повторения при всех значениях параметра внутреннего повторения (в данном случае b). В рассмотренном примере тело повторения (команды ВВЕСТИ, $S = a * b$, ВЫДАТЬ), внутреннее повторение предписывает вычисление площади прямоугольника $S = a * b$, для одного значения a и b . При этом при переходе от повторения к повторению значения a и b будут чередоваться в следующем порядке:

```
a 1 1 1 2 2 2 3 3 3
b 1 2 3 1 2 3 1 2 3
```

Таким образом, в рассмотренном примере тело внутреннего цикла будет выполнено 9 раз.

Комментарии и пояснения к алгоритму. Комментарии и пояснения к алгоритму могут помещаться между командами в любом месте алгоритма. Начало и конец алгоритма обозначаются звездочками (например, * комментарий *). Команда, следующая за комментарием, помещается с новой строки.

4.1.2. ТАБЛИЦЫ

При выполнении вычислений часто приходится иметь дело с данными, размещаемыми в таблицах. В алгоритмическом языке таблицей мы будем называть конструкцию, состоящую из m одинаковых строк и n одинаковых столбцов, в которых размещены числовые данные — элементы таблицы. Такую таблицу называют еще прямоугольной матрицей $m \times n$. Положение каждого элемента таблицы строго определено номером строки и номером столбца, которые принято называть индексами элемента. Каждая таблица обозначается своим именем и размерностью. Например, таблица A , имеющая m строк и n столбцов, описывается так: $A [1 : m, 1 : n]$. Обозначения $1 : m$ и $1 : n$, указывающие начальные и конечные номера строк и столбцов, принято называть граничными парами.

Элементы таблицы могут рассматриваться как переменные, имеющие общее имя и различающиеся по индексам. Поэтому пере-

менные, входящие в состав таблиц, называют переменными с индексами и обозначают так: $A[1, 1]$, $B[3, 5]$, $C[i, j]$, $E[m, n]$. При этом первый индекс определяет номер строки, а второй — номер столбца. Переменные с индексами могут входить в состав выражений и включаться в соответствующие команды.

В частном случае таблица может состоять из одной строки или одного столбца. Такую одномерную таблицу называют вектором. В этом случае размерность таблицы будет характеризоваться одной граничной парой (например, $B[1 : n]$), а переменные, входящие в таблицу — одним индексом. Например, $B[1], B[2], \dots, B[i], \dots, B[n]$.

Если в алгоритме предусматривается обработка таблиц, то в перечни аргументов, результатов (а в случае необходимости и в список промежуточных величин) включают имена таблиц с их размерностями, а перед списком таблиц помещают служебное слово таб. Например `арг R1 = p, таб A [1 : m, 1 : n], CB [1 : 5]`.

В ПМК практически нет возможности размещения в памяти всех элементов таблиц, поскольку объем памяти в ПМК сильно ограничен. Поэтому значения переменных, входящих в состав таблиц, вводят и обрабатывают на ПМК последовательно, выделяя для ввода и хранения переменных один регистр памяти. Этот регистр указывают перед именем таблицы. Например, таб $RX = A[1 : n]$.

Число строк в таблице будем называть ее длиной, а число столбцов — шириной. Одномерные таблицы будем характеризовать только их длиной. Часто бывает, что при составлении алгоритма неизвестна длина таблицы и/или ее ширина. Они становятся известными непосредственно перед вычислениями или даже в процессе вычислений. Такие таблицы называют таблицами неопределенной длины (ширины). Соответствующие граничные пары в квадратных скобках для таких таблиц заменяют звездочками. Например, $M[* , 1 : n]$. Для одномерных таблиц граничную пару вместе с квадратными скобками можно опустить.

В тексте алгоритма длина и ширина таблицы неопределенной длины рассматриваются как значения некоторых переменных длин и шир, аргументами которых служат имена таблиц. Эти переменные обозначаются длин (M), шир (A). Слова длин и шир являются служебными, а M и A — именами таблиц.

4.2. ПРИМЕРЫ АЛГОРИТМОВ ТИПОВЫХ ВЫЧИСЛЕНИЙ

Рассмотрим алгоритмы часто встречающихся ручных вычислений, в которых используются табличные данные и стековая память ПМК. В левой части листа будем помещать алгоритм вычислений, а в правой — команды ПМК, необходимые для реализации команд присваивания алгоритма.

Последовательности чисел равной, но неопределенной длины рассматриваются здесь как одномерные таблицы. Аналогично предыдущему примеру в каждом цикле выдается текущее значение суммы произведений S_i .

Примеры:

1. Расчет сопротивления электрической цепи, состоящей из n параллельных ветвей.

Дано: Электрическая цепь, состоящая из n параллельных ветвей, с сопротивлением каждой из них r_i .

Требуется: Найти общее сопротивление цепи $R_{\text{общ}} = 1 / \sum_{i=1}^n 1/r_i$.

Обозначим $S = \sum_{i=1}^n 1/r_i$, $S_i = S_{i-1} + 1/r_i$, $S_0 = 0$.

АЛГОРИТМ 4.1

алг сопротивление цепи

арг таб $RX = r$

рез $RX = R_{\text{общ}}$

нач $i, RX = S, RY = S$

$S = 0$

КОМАНДЫ ПМК

НАЖАТЬ

00. Cx (0)

01. B! (0, 0)

для i от 1 до длин (r)

нц

ВВЕСТИ $r[i]$ ($r[i], S$)

$S = S + 1/r[i]$

кц

ВЫДАТЬ S

$R_{\text{общ}} = 1/S$

кон

НАЖАТЬ

02. F1/x ($1/r[i], S$)

03. + ($S + 1/r[i]$)

04 F1/x ($R_{\text{общ}}$)

2. Вычисление суммы квадратов чисел.

Дано: Последовательность чисел $a = a_1, a_2, \dots, a_i \dots$.

Требуется: Вычислить сумму квадратов этих чисел $S = \sum_{i=1}^t a_i^2$.

КОМАНДЫ ПМК

АЛГОРИТМ 4.2

алг сумма квадратов

арг таб $RX = a$

рез $RX = S$

нач $i, RY = S$

$S = 0$

НАЖАТЬ

00. Cx (0)

01. B! (0, 0)

для i от 1 до длин (a)

нц

ВВЕСТИ $a[i]$ ($a[i], S$)

$S = S + a[i] * * 2$

кц

ВЫДАТЬ S

кон

02. Fx² ($a[i] * * 2, S$)

03. + ($S + a[i] * * 2$)

3. Суммирование последовательности чисел (строки, столбца).

Дано: Последовательность чисел $a = a_1, \dots, a_i$.

Требуется: Найти сумму последовательности чисел

Упражнение

$$S = \sum_{t=1}^i a_t.$$

Обозначим: $S_i = S_{i-1} + a$. Положим $S_0 = 0$.

КОМАНДЫ ПМК

АЛГОРИТМ 4.3

```

алг сумма столбца
арг таб RX = a
рез RX = S
нач i, RY = S
S = 0
    НАЖАТЬ
    00. Сх (0)
    01. В! (0, 0)

для i от 1 до длин (a)
нц
    ВВЕСТИ a [i] (a [i])
    S = S + a [i]
    НАЖАТЬ
    02.+ (S)
    ВЫДАТЬ S
кц
кон

```

После каждого ввода a и нажатия клавиши $+$ выдается значение текущей суммы S_i . После исчерпания списка чисел текущая сумма S_i становится окончательной S . Для текущего индекса i регистр памяти не выделяется. В этом нет необходимости, поскольку значение i в память ПМК не вводится.

4. Вычисление суммы произведений чисел:

$$S = \sum_{i=1}^i a_i b_i.$$

Дано: Последовательность пар чисел a_i, b_i ($i = 1, 2, \dots, i$).
Требуется: Вычислить сумму произведений пар чисел

$$S = \sum_{i=1}^i a_i b_i.$$

Решение. $S_i = S_{i-1} + a_i b_i$ ($S_0 = 0$).

АЛГОРИТМ 4.4

КОМАНДЫ ПМК

```

алг сумма произведений
арг таб RX = a, RY = b
рез RX = S
нач i, RY = S, RZ = S
S = 0
    НАЖАТЬ
    00. Сх (0)
    01. В! (0, 0)
    02. В! (0, 0, 0)

для i от 1 до длин (a)
нц
    ВВЕСТИ a [i]! b [i] (a [i], b [i], S)
    НАЖАТЬ
    03. × (a [i] * b [i], S)
    04. + (S + a [i] * b [i], S)
    ВЫДАТЬ S
кц
кон

```

4.2.1. Составить алгоритмы:

- 1) Произведения n сомножителей.
- 2) Вычисления площади круга для n значений диаметра D .
- 3) Вычисление факториала $N = n!$ (исключая $n = 0$).

4.3. ПРИМЕР РЕШЕНИЯ ЗАДАЧИ С ПРЕДСТАВЛЕНИЕМ РЕЗУЛЬТАТОВ В ВИДЕ ТАБЛИЦЫ

Рассмотрим пример составления алгоритма более сложной задачи с повторяющимися вычислительными операциями. Это типичный пример часто встречающихся расчетов с представлением результатов в табличной форме. Решение подобных задач с представлением результатов в виде многострочных и многографных таблиц требует точной постановки задачи и разработки алгоритма ее решения. Алгоритм составляется как для ручного, так и для автоматического (программируемого) режима вычислений. При этом, как будет показано в следующей главе, алгоритмы вычислений в обоих режимах мало различаются.

Решение с помощью ПМК достаточно сложных задач с много-кратно повторяющимися вычислительными операциями включает в себя следующие этапы:

1. Постановка задачи.
2. Разработка алгоритма решения задачи.
3. Решение задачи по разработанному алгоритму.

В качестве примера рассмотрим задачу о наложении двух синхронных гармонических колебаний [10]. На этом примере рассмотрим все перечисленные этапы решения задачи на ПМК.

4.3.1. ПОСТАНОВКА ЗАДАЧИ

С учетом использования ПМК постановка задачи может быть подразделена на три этапа:

1. Формулировка задачи.
2. Анализ сформулированной задачи.
3. Разработка бланка таблицы, заполняемого при проведении расчетов.

Формулировка задачи. Определяется, что дано и что требуется получить в результате решения задачи.

Дано: Два гармонических синхронных колебания:

$$s_1 = A_1 \cos(\omega t - \psi_1); s_2 = A_2 \cos(\omega t - \psi_2),$$

путем наложения (суперпозиции) образующие результирующее колебание

$$s = A \cos(\omega t - \phi).$$

Требуется: Рассчитать значения амплитуды A и фазы ϕ результирующего гармонического колебания. Расчет произвести

для N значений сдвига фаз и L значений соотношения амплитуд: $n = A_2/A_1$.

Анализ сформулированной задачи. На этом этапе уясняется формулировка задачи, определяются параметры и особенности вычислительного процесса, реализующего сформулированную задачу, а также устанавливаются возможные пути ее решения. В процессе анализа сформулированной задачи может быть рассмотрен следующий примерный круг вопросов:

уяснение формулировки задачи;

определение основных расчетных формул (если они заданы) или их уяснение и уточнение;

определение области допустимых значений аргументов и порядка их чередования;

возможность упрощения расчетов, например, путем преобразования расчетных формул и выделения в них постоянных величин параметров и констант;

особые точки вычисляемых функций (например, точка разрыва непрерывности);

выбор пути решения задачи.

Для рассматриваемого примера вычисления амплитуды и фазы результирующего колебания это будут следующие вопросы:

1. Определение основных расчетных формул для вычисления амплитуды и фазы результирующего колебания по условию $s_1 = -A_1 \cos(\omega t - \psi_1)$, $s_2 = A_2 \cos(\omega t - \psi_2)$.

В результате сложения двух колебаний, т. е. $s = s_1 + s_2$, получим

$$s = A_1 \cos(\omega t - \psi_1) + A_2 \cos(\omega t - \psi_2),$$

или, выполнив простые тригонометрические преобразования, придем к формуле

$$s = (A_1 \cos \psi_1 + A_2 \cos \psi_2) \cos \omega t + (A_1 \sin \psi_1 + A_2 \sin \psi_2) \sin \omega t. \quad (4.1)$$

Обозначим:

$$A_1 \cos \psi_1 + A_2 \cos \psi_2 = A \cos \varphi;$$

$$A_1 \sin \psi_1 + A_2 \sin \psi_2 = A \sin \varphi. \quad (4.2)$$

Подставляя (4.2) в (4.1), получим

$$s = A (\cos \varphi \cos \omega t + \sin \varphi \sin \omega t),$$

или

$$s = A \cos(\omega t - \varphi). \quad (4.3)$$

Формула (4.3) описывает результирующий колебательный процесс с амплитудой A и фазой φ . Для нахождения A и φ обратимся к выражению (4.2). Возведя каждое из уравнений в квадрат и сложив их, получим:

$$A^2 = A_1^2 + A_2^2 + 2A_1 A_2 \cos(\psi_1 - \psi_2),$$

а разделив первое уравнение на второе, определим

$$\operatorname{tg} \varphi = \frac{A_1 \sin \psi_1 + A_2 \sin \psi_2}{A_1 \cos \psi_1 + A_2 \cos \psi_2}.$$

В результате будут получены расчетные формулы для A и φ :

$$A = \sqrt{A_1^2 + A_2^2 + 2A_1 A_2 \cos(\psi_1 - \psi_2)}; \quad (4.4)$$

$$\varphi = \operatorname{arctg} \frac{A_1 \sin \psi_1 + A_2 \sin \psi_2}{A_1 \cos \psi_1 + A_2 \cos \psi_2}.$$

2. Возможность упрощения вычислений. Обозначим $A_2/A_1 = n$ — соотношение амплитуд исходных колебаний и $\psi_1 - \psi_2 = \varphi$ — сдвиг фаз между ними. Тогда в результате преобразования (4.4) получим

$$A = A_1 \sqrt{1 + n^2 + 2n \cos \varphi}; \quad (4.5)$$

$$\varphi = \operatorname{arctg} \frac{\sin \psi_1 + n \sin \psi_2}{\cos \psi_1 + n \cos \psi_2}.$$

Если первое колебание считать опорным, т. е. таким когда $A_1 = 1$, а $\psi_1 = 0$, то расчетные формулы (4.5) можно упростить:

$$A = \sqrt{1 + n^2 + 2n \cos \varphi}; \quad (4.6)$$

$$\varphi = \operatorname{arctg} ((n \sin \varphi)/(1 + n \cos \varphi)).$$

3. Уяснение расчетных формул. Что следует понимать под A , n и φ . Поскольку опорным колебанием в формулировке задачи выбрано колебание $s_1 = A_1 \cos(\omega t - \psi_1)$, в котором $A_1 = 1$, а $\psi_0 = 0$, то $n = A_2/A_1 = A_2$, т. е. значение n показывает, во сколько раз амплитуда второго колебания больше или меньше первого.

Значение амплитуды результирующего колебания A показывает, во сколько раз она больше или меньше амплитуды опорного колебания A_1 (A — величина безразмерная).

Фаза результирующего колебания φ показывает значение сдвига фаз в радианах между фазой результирующего колебания s и фазой опорного колебания s_1 .

4. Тип вычислительного процесса. Тип вычислительного процесса — циклический с двумя циклами по ψ_i и n_j . Порядок чередования аргументов в процессе вычислений (внешний цикл по ψ_i , внутренний по n_j):

$$\begin{array}{ccccccc} \psi_1 & n_1 & \dots & n_j & \dots & n_L \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \psi_i & n_1 & \dots & n_j & \dots & n_L \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \psi_N & n_1 & \dots & n_j & \dots & n_L \end{array}$$

5. Особые точки вычисляемых функций. Из формулы (4.6) легко заметить, что при вычислении фазы результирующего колебания невозможно вычисление дроби под знаком арктангенса при $1 + n \cos \psi = 0$. Аргумент арктангенса в этом случае будет равен бесконечности, а фаза результирующего колебания $\varphi = \arctg \infty = \pi/2$. Следовательно, если окажется, что $1 + n \cos \psi = 0$, то не следует вычислять значение φ . Нужно сразу положить его равным $\pi/2$.

Разработка бланка таблицы. Бланк, заполняемый в процессе решения задачи входными, промежуточными и выходными данными, представляет собой таблицу, в которой заранее записаны номера и наименования граф, а над таблицей слева указаны наименования и значения параметров вычисляемых функций. Количество граф в таблице устанавливается в каждом конкретном случае отдельно. В нулевой граfe всегда помещается порядковый номер строки, а в первой — аргумент — параметр внутреннего цикла. Количество строк в таблице также определяется в каждом конкретном случае отдельно. В частном случае, если вычислительный процесс линейный, то таблица будет состоять из одной строки. В общем случае, когда вычислительный процесс носит циклический характер, необходимо прежде всего решить, где, в каких графах поместить независимые переменные, являющиеся параметрами циклов. Здесь возможны следующие варианты:

1. Над таблицей слева помещаются наименование и значение параметров внешнего цикла, которое для всего листа бланка сохраняется постоянным. Параметр внутреннего цикла помещается при этом в первой граfe таблицы и изменяется от строки к строке (табл. 4.1).

2. Параметр внешнего цикла записывается в первой граfe и сохраняется постоянным во всех строках, пока не будут перебраны все значения параметра внутреннего цикла, помещенного во второй граfe. После этого записывается второе значение внешнего параметра цикла в первой граfe табл. 4.2.

Таблица 4.1

Вариант 1

$n = n_j$		
Ψ	$A = \sqrt{1 + n^2 + 2n \cos \psi}$	$\varphi = \arctg \frac{n \sin \psi}{1 + n \cos \psi}$
Ψ_1		
\dots		
Ψ_i		
\dots		
Ψ_N		

Лист ...

3. Значения параметров внешнего цикла записываются в графах таблицы, а внутреннего — в строках первой ее графы. Этот вариант используется при расчетах таблиц относительно небольшого объема (табл. 4.3).

Первый вариант используется при больших объемах вычислений, когда необходимо заполнить бланки на многих листах, третий — при небольших объемах (один лист). Второй вариант является промежуточным между первым и третьим и чаще всего используется для расчетов на ПМК.

Варианты бланков для расчета амплитуды и фазы результирующего колебания приведены в табл. 4.1—4.3. Для представления результатов в рассматриваемом примере выберем третий вариант таблицы, поскольку число вычисляемых значений должно быть небольшим. Заполнение табл. 4.3 в процессе вычислений будет носить циклический характер и включать в себя два цикла: один внешний — по индексу i (значения Ψ изменяются при переходе от одной строки таблицы к другой), а второй внутренний — по индексу j (значения n изменяются внутри строки при переходе от вычисления одной пары значений A и φ к другой).

Таблица 4.2

Вариант 2

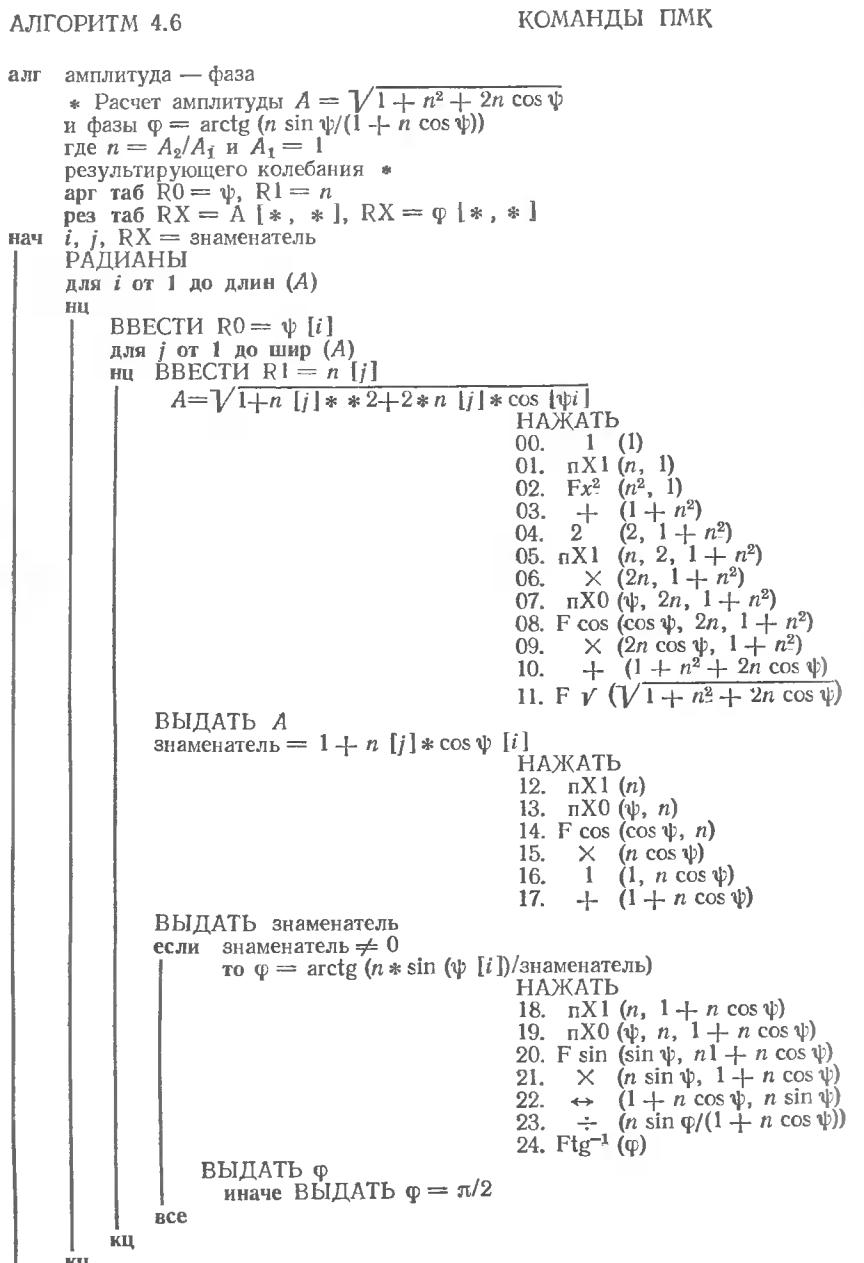
Ψ	n	$A = \sqrt{1 + n^2 + 2n \cos \psi}$	$\varphi = \arctg \frac{n \sin \psi}{1 + n \cos \psi}$
Ψ_1	n_1		
	\dots		
	n_L		
Ψ_2	n_1		
	\dots		
	n_L		
\dots	\dots		
Ψ_N	n_1		
	\dots		
	n_L		

Таблица 4.3

Вариант 3

Ψ	n_i		n_j		n_L	
	A	φ	A	φ	A	φ
1	2	3	...	2j	2j+1	...
Ψ_1						
\dots						
Ψ_N						

4.8.2. РАЗРАБОТКА АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ



КОМАНДЫ ПМК

Расчет амплитуды A и фазы φ

ψ	$n = 1$		$n = 2$	
	A	φ	A	φ
0	2	0	3	0
$\pi/2$	1.4142135	0.7853981	2.2360679	1.1071486
π	0	$\pi/2$	1	0
$3\pi/2$	1.4142135	-0.7853982	2.2360679	-1.1071487
2π	2	0	3	0

Выполним расчет таблицы по разработанному алгоритму для $\psi = 0, \pi/2, \pi, 3\pi/2, 2\pi$ и $n = 1; 2$. Результаты решения сведем в табл. 4.4 (бланк — см. табл. 4.3, вариант 3).

Упражнения

4.3.1. Разработать алгоритмы расчета на ПМК значений надежности P системы, состоящей из n одинаковых параллельно соединенных элементов, каждый из которых обладает надежностью p . Расчет производится по формуле

$$P = 1 - (1 - p)^n.$$

По составленному алгоритму решить задачу для $p = 0.8, 0.95$ и $n = 1, 2, 3$.

4.3.2. Разработать алгоритм расчета на ПМК таблицы функций, определяемой формулой

$$y = \frac{A + Bx}{(A + x)(B + x)}$$

при $A = \text{const}$, $B = \text{const}$ и $x = x_1, \dots, x_n$. По составленному алгоритму решить задачу для $A = 7.87$; $B = 10.75$ и $x = 1; 3; 5$.

Раздел III

ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ В ПРОГРАММИРУЕМОМ РЕЖИМЕ

Глава 5. СОСТАВЛЕНИЕ ЛИНЕЙНЫХ ПРОГРАММ

До сих пор рассматривались вычисления на ПМК, производимые вручную. Начиная с настоящей главы обратимся к автоматизации указанных вычислений. В отличие от ЭВМ на ПМК вычисления автоматизируются не полностью. Операции входа исходных данных и выдачи результатов здесь производятся вручную. Введенная в память ПМК программа обеспечивает автоматическую работу лишь от одного останова, предусмотренного программой, до другого. Во время пауз алгоритмом могут быть предусмотрены ручные вычисления, связанные с необходимостью преобразования исходных данных и результатов, а также другие действия пользователей, связанные с управлением вычислительным процессом. Алгоритм вычислительного процесса при этом будет представлять собой последовательность чередующихся частей, выполняемых пользователем вручную и автоматически по введенной в ПМК программе. Такой режим вычислений будем называть не автоматическим, а программируемым, подчеркивая тем самым использование программы для частичной автоматизации вычислений.

Алгоритмы вычислений в программируемом режиме различаются между собой степенью автоматизации, т. е. долей автоматически выполняемых операций из общего их количества. В настоящей главе будут рассмотрены вычислительные процессы, в которых автоматизированы только линейные участки алгоритма. Ветвления и повторения осуществляются при этом вручную. Для выполнения вычислений, в которых автоматизированы линейные участки алгоритма, необходимо прежде всего уметь составлять линейные программы для ПМК.

Программой принято называть алгоритм, записанный в форме, воспринимаемой и выполняемой ЭВМ или ПМК. Язык, предназначенный для записи программ и данных к ним, принято называть языком программирования, которое понимается в узком и широком смысле слова. В узком смысле оно сводится к переводу алгоритма, ориентированного на пользователя и не воспринимаемого машиной, на язык программирования, т. е. к перекодировке алгоритма в программу, непосредственно воспринимаемую машиной. В широком смысле программирование объединяет всю совокупность работ, необходимых для создания и эксплуатации программ для ЭВМ или ПМК. Его можно разделить на три относительно самостоятельных процесса:

1. Разработка алгоритма и программы решения задачи.
2. Решение с помощью ПМК задачи с конкретным вариантом исходных данных.
3. Сопровождение алгоритма и программы в процессе их эксплуатации.

Разработка алгоритма и программы решения задачи на ПМК включает в себя следующие этапы:

- 1) Постановка задачи.
- 2) Разработка алгоритма.
- 3) Разработка программы.
- 4) Ввод программы в память ПМК и ее отладка.
- 5) Контрольное решение задачи.
- 6) Оформление документации.

Решение задачи с помощью ПМК включает в себя следующие этапы:

- 1) Ввод программы в программную память ПМК.
- 2) Выполнение контрольного решения.
- 3) Решение задачи с требуемыми исходными данными.

Сопровождение алгоритма и программы можно рассматривать как продолжение их разработки, но уже в процессе эксплуатации. Процесс сопровождения может включать в себя выполнение следующих работ:

1) Определение пригодности ранее разработанной программы к изменявшимся условиям задачи и внесение в них с этой целью необходимых изменений и дополнений.

2) Устранение ошибок в алгоритме и программе, обнаруженных в процессе эксплуатации (неточности в первоначальной постановке задачи, отсутствие формулировки ограничений, неявно заложенных в алгоритме и программе, а также ошибки в программе, выявленные в процессе ее эксплуатации при определенных соотношениях исходных данных, не предусмотренных контрольным решением).

3) Выполнение контрольного решения после внесения изменений в алгоритм и программу.

4) Корректировка документации на программу.

В предлагаемой главе будут изложены все сведения, необходимые пользователю ПМК для решения задач по готовым программам.

Порядок подготовки и решения задач в программируемом режиме на примере вычисления амплитуды и фазы результирующего колебания, алгоритмы расчета которых осуществляются в ручном режиме (алгоритм 4.6), был приведен в предыдущей главе (п. 4.3). Формулировка задачи и форма бланка таблицы для заполнения результатами расчета остаются при этом без изменений. Поэтому рассмотрение порядка разработки алгоритма и программы начнем со второго ее этапа — разработки алгоритма для программируемого режима работы ПМК.

5.1. РАЗРАБОТКА АЛГОРИТМА

В программируемом режиме так же, как и в ручном, целью этапа разработки алгоритма является составление плана или проекта вычислительного процесса, подлежащего выполнению с помощью ПМК. Вместе с тем разработка алгоритма в программируемом режиме имеет свои особенности. В процессе ее производится:

сегментация алгоритма с выделением из его состава частей, выполняемых вручную и автоматически на ПМК по введенной программе;

описание частей алгоритма, выполняемых вручную, с помощью алгоритмического языка.

Автоматически выполняемые части алгоритма должны быть ориентированы не на ПМК, а на пользователя, на понимание им сути вычислительного процесса. Вместе с тем они должны быть описаны в такой форме, которая позволяла бы производить однозначное и формальное преобразование указанных частей алгоритма в соответствующие последовательности команд ПМК. Удовлетворение этих противоречивых требований в одном алгоритме вряд ли возможно. Ориентированный на ПМК алгоритм становится непонятным для пользователя. Но алгоритм, ориентированный на пользователя, в большинстве случаев не обеспечивает формального преобразования его в команды программы. Все это вызывает необходимость в многоуровневой разработке алгоритма и программы. На первом уровне разрабатывается алгоритм, ориентированный на пользователя с помощью лексикона (ранее рассмотренного алгоритмического языка) [14]. Первый уровень алгоритма представляет собой как бы общий план решения задачи. Команды указанного алгоритма условимся называть А-командами. На втором уровне алгоритма эти команды детализируются, заменяются, как правило, несколькими командами алгоритмического языка второго уровня, которые условимся называть Б-командами. На втором уровне детализируются не все А-команды, а только те из них, которые подлежат автоматическому выполнению на ПМК. На третьем уровне разработки алгоритма каждая Б-команда заменяется по формальным правилам последовательностью команд программы. Алгоритм на третьем уровне его представления — это программа, непосредственно воспринимаемая и выполняемая ПМК. Команды программы в отличие от А-команд и Б-команд будем называть П-командами.

Каждый из трех уровней представления алгоритма требует для своего описания соответствующих языковых средств. Для описания алгоритма на первом уровне будем использовать рассмотренный выше алгоритмический язык [14]. Для описания алгоритма на втором уровне будем использовать язык, близкий к широко распространенному в мире языку программирования БЕЙСИК (BASIC — Beginner's ALL-purpose Symbolic Instruction

Code — многоцелевой язык символьических команд для начинающих).

Этот язык будет использоваться не в качестве языка программирования, а в качестве языка описания алгоритмов, ориентированного на упрощение процесса ручного преобразования их команд в соответствующие команды программы. Будем называть этот язык БЕЙСИК-ПМК.

Наконец, для описания алгоритма на третьем уровне программы для ПМК будем пользоваться языком клавишного набора — входным языком программирования ПМК.

5.1.1. АЛГОРИТМИЧЕСКИЙ ЯЗЫК БЕЙСИК-ПМК

Поскольку алгоритмический язык БЕЙСИК-ПМК не является языком программирования, то команды этого языка не могут быть введены непосредственно в память и выполнены ПМК. Однако в большинстве случаев разработать и осмыслить алгоритм на БЕЙСИК-ПМК намного проще, чем сразу составлять программу. После того как БЕЙСИК-алгоритм разработан, преобразование его команд в команды программы не вызовет особых трудностей.

Общий вид алгоритма на БЕЙСИК-ПМК. Алгоритм на БЕЙСИК-ПМК состоит из последовательности строк. Каждая строка начинается с номера строки (целое число). Правее номера строки помещается Б-команда. Строки не обязательно нумеровать по порядку. Важно лишь, чтобы они нумеровались в возрастающем порядке: 5, 7, 10, 20 ... и т. д. Условимся возрастающий порядок номеров строк обозначать в конструкциях БЕЙСИК-ПМК $n+0, n+1, n+2 \dots$ и т. д. Тогда общий вид алгоритма на БЕЙСИК-ПМК будет выглядеть так:

```
n+0 команда 1  
n+1 команда 2  
...  
n+i команда i  
...  
n+l команда l  
n+l+1 STOP
```

Последней командой алгоритма на БЕЙСИК-ПМК является команда STOP, предписывающая останов вычислений. В языке БЕЙСИК-ПМК, как и в базовом языке БЕЙСИК, используются только буквы латинского алфавита и английские ключевые слова, определяющие названия Б-команд.

Константы и переменные. Как и в рассмотренном ранее алгоритмическом языке, так и в БЕЙСИК-ПМК используемые величины делятся на переменные и постоянные константы, которые в естественной форме представления записываются так же, как ранее нами записывались в алгоритмическом языке: — 12.5, 3.1415926 и т. п. В показательной форме константы записываются в виде мантиссаЕпорядок. Например 0.25E14, —1.25E—3, 563E—03. Порядок может быть только целым, состоящим не более чем из двух цифр.

Именем переменной в языке БЕЙСИК-ПМК служит имя регистра, состоящее из буквы R и его номера (адреса). Имя регистра является внутренним именем переменной, устанавливаемым на втором уровне представления алгоритма (в отличие от внешних имен, используемых на первом уровне представления). В качестве имен переменных в БЕЙСИК-ПМК могут использоваться не только имена регистров адресуемой памяти (R0 ... R9, RA, RB, RC, RD, RE), но и имена регистров стековой памяти (RX, RY, RZ, RT), хотя правила использования последних приходится оговаривать особо (см. команда присваивания).

Выражения. Как и в алгоритмическом языке первого уровня, в БЕЙСИК-ПМК из констант, а также имен переменных (регистров) с помощью знаков операций и круглых скобок строятся выражения. В отличие от алгоритмического языка первого уровня в выражениях БЕЙСИК-ПМК могут использоваться в выражениях только имена регистров, начинающиеся с буквы R. Все выражения записываются только в одну строку без надстрочных и подстрочных символов. Не допускается использование греческих и малых латинских букв. Знаки операций используются те же, что и в алгоритмическом языке первого уровня, а именно + - / * (умножить), ** (возвести в степень). Имена стандартных функций также записываются прописными латинскими буквами. Обратные тригонометрические функции обозначаются сокращенно ASIN, ACOS, ATG (вместо arcsin, arccos, arctg на первом уровне представления). Полный перечень используемых в БЕЙСИК-ПМК функций приведен в прил. 1. Примеры выражений:

$$2 * \text{SIN}(X) * \text{COS}(X), \quad RX * \text{ABS}(A - B), \quad R1 * R1 * * 2/4.$$

Команда присваивания. Общий вид команды:
номер строки_регистр=БЕЙСИК-выражение [; (стек)]

Примеры команд присваивания:

1) А-команда:

$$M = (1/\sqrt{3} + 3\sqrt{5}) - \text{arctg}(1.2) * 10^{-3}$$

Б-команда:

$$1 \quad RX = (1/\text{SQR}(3) + 3 * \text{SQR}(5) - \text{ATG}(1.2)) * 1E - 3$$

2) А-команда:

$$P = \sin^4 \varphi + \cos^4 \varphi - (1 - \tan^2 \varphi) / \tan^2 \varphi$$

Б-команда:

$$10 \quad RX = \text{SIN}(R1) * * 4 + \text{COS}(R1) * * 4 - (1 - \text{TG}(R1) * * 2) / \text{TG}(R1) * * 2$$

3) А-команда:

$$S = \pi D^2 / 4$$

Б-команда:

$$12 \quad R2 = R1 * R1 * * 2/4$$

4) А-команда:

$$x[i] = x[i] + 1$$

Б-команда:

$$15 \quad RA = RA + 1$$

Команда останова вычислений. На первом уровне алгоритма эта команда отсутствует (используется только на втором и третьем уровнях). Общий вид команды:

Б-команда:
номер STOP [(стек)]

П-команда (команда ПМК):
адрес.—С/П [(стек)]

Команда STOP предписывает осуществить останов вычислений, выполняемых на ПМК в автоматическом режиме. БЕЙСИК-алгоритм и программа для ПМК всегда заканчиваются командой STOP (С/П). Однако эта команда может использоваться и внутри алгоритма программы. В последнем случае она предписывает остановить вычисления до окончания работы программы, что бывает необходимым по ходу решения задачи. Пуск программы после останова, выполненного по команде STOP (С/П), осуществляется пользователем вручную путем нажатия клавиши С/П, что предписывается в алгоритме первого уровня командой НАЖАТЬ.

5.1.2. ОСОБЕННОСТИ РАЗРАБОТКИ АЛГОРИТМА С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА БЕЙСИК-ПМК

Порядок записи алгоритмов. Как уже отмечалось в начале параграфа, алгоритм решения задачи на ПМК может рассматриваться как совокупность трех уровней его представления. Каждый последующий уровень является детализацией предыдущего, а последний (третий) уровень представляет собой программу решения задачи, подлежащей вводу и автоматическому выполнению на ПМК. Для удобства восприятия алгоритмы всех трех уровней целесообразно записывать на одном листе бумаги, разделив его на три части. В левой части листа будем начинать записывать А-команды алгоритмического языка, в средней — команды БЕЙСИК-алгоритма, а в правой — команды программы. Однако поместить каждый уровень представления алгоритма в отдельную колонку на стандартном листе бумаги шириной 203 мм не удастся. Чтобы можно было разместить алгоритмы всех трех уровней на одном листе бумаги, форму записи алгоритма несколько изменим. Если потребуется, команды алгоритма первого уровня (А-команды) будем записывать по всей ширине листа, команды алгоритма второго уровня (Б-команды) — правее примерно на 15 символов от начала строки соответствующей команды первого уровня, а команды программы — примерно на 15 символов правее соответствующих команд алгоритма второго уровня.

При такой записи алгоритмы каждого из уровней будут достаточно четко отделяться друг от друга, а их команды — различаться не только расположением относительно начала строки, но и другими внешними признаками. Служебные слова в А-командах даются на русском языке, в Б-командах — на английском (прописными буквами). А-команды не имеют номера, а Б-команды пронумерованы и их номера отличаются от адресов команд программы тем, что первые отделяются от самой команды пробелом, а вто-

рые — точкой. Если строки не хватит для записи команды на каком-либо уровне, то ее продолжают на следующей строке уступом на 3—5 позиций правее начала команды.

Команды программы записываются в правой трети листа столбцом, как это делалось ранее. Состояние стека после выполнения каждой команды обязательно приводится в круглых скобках. Переменные (как одиночные, так и в выражениях), характеризующие состояние стека и помещаемые в круглых скобках, могут быть как внешними, так и внутренними, используемыми в БЕЙСИК-ПМК. Знаки математических операций и указатели стандартных функций в выражениях, стоящих в скобках, могут быть как общепринятыми, так и предусмотренными в БЕЙСИК-ПМК. Для более компактной записи состояния стека вместо записи выражений допускается давать ссылки на них, записанные ранее и отражающие состояние регистра X в предыдущих командах. Например, $6RX$ означает ссылку на выражение, записанное в Б-команде 6, если она помещена в БЕЙСИК-алгоритме, или на состояние регистра X после выполнения команды программы с адресом 06.

Использование регистров стековой памяти в БЕЙСИК-алгоритмах. Характерной особенностью БЕЙСИК-алгоритмов является широкое использование результата предыдущего действия, рассматриваемого как значение некоторой текущей переменной. В качестве имен текущих переменных используются имена регистров стековой памяти RX , RY , RZ , RT . Наиболее часто в качестве имени текущей переменной используется RX . В результате выполнения операции присваивания вычисленное значение выражения правой части всегда помещается в регистр X, независимо от того, какая переменная помещалась в левой части. Таким образом, после выполнения любой команды присваивания ее результатом является значение переменной RX . В ходе выполнения алгоритма содержимое регистра X (а значит, и переменная RX) постоянно меняется, перемещаясь в другие регистры стека. Для определенности будем полагать, что значение RX , вычисленное в предыдущей команде присваивания, сохраняется неизменным до момента окончания текущей команды, когда полученное значение выражения становится новым значением RX и помещается в регистр X, а старое помещается в регистр Y.

Рассмотрим последовательность трех команд присваивания и установим, как при этом будут изменяться значения переменных RX , RY , RZ .

$A = p$

$$1 RX = R3; (IRX)$$

$$q = 1 + n \cos \psi$$

$$2 RX = 1 + R1 * \cos(R2); (2RX, IRX)$$

$$00. \pi X3(R3)$$

$$01. 1(1, R3)$$

$$02. \pi X1(R1, 1, R3)$$

$$03. \pi X2(R2, R1, 1, R3)$$

$$04. F \cos(\cos(R2), R1, 1, R3)$$

05. $\times (R1 * \cos(R2), 1, R3, R3)$
 06. $+ (1 + R1 * \cos(R2), R3, R3, R3)$

$$s = n * \sin \Phi/q$$

- 3 $RX = R1 * \sin(R2)/RX; (3RX, 2RX, 1RX)$
 07. $\pi X1(R1, R1 * \cos(R2), 1, R3)$
 08. $\pi X2(R2, R1, 05RX, 1, R3)$
 09. $F \sin(\sin(R2), R1, 05RX, 1, R3)$
 10. $\times (R1 * \sin(R2), 05RX, 1, R3)$
 11. $\leftrightarrow (05RX, 09RX, 1, R3)$
 12. $\div (09RX/05RX, 1, R3)$

В результате выполнения первой команды присваивания в регистре X будет помещено значение IRX . После второй команды присваивания в регистре X будет записано значение $2RX = 1 + R1 * \cos(R2)$, а в регистре Y — $1RX$. После третьей команды присваивания значение $1RX$ переместится в регистр Z, а $2RX$ — в регистр Y. При этом после выполнения каждой команды в регистре X каждый раз будет помещаться новое значение. Как видим, состояние стека после выполнения Б-команд определяется по тем же правилам, что и после выполнения команд программы. Иными словами, команды БЕЙСИК-алгоритма можно рассматривать как укрупненные команды программы ПМК. Однако это утверждение справедливо лишь в том случае, если при выполнении команд стек не переполняется. Если рассматривать процесс вычисления выражения, стоящего в правой части команды присваивания, как ввод значений переменных и получение промежуточного результата, затем новый ввод переменных и получение нового промежуточного результата с использованием старого и т. д., то стек не будет переполняться при условии, что число вводимых переменных для получения промежуточного результата будет не более трех. При этом в число переменных включается также используемый предыдущий результат.

Состояние стека после выполнения любой БЕЙСИК-команды можно изменить с помощью команд присваивания с переменными RX , RY , RZ и выполнить следующие операции:

1. Восстановление содержимого регистра X после выполнения команды присваивания. Пусть после выполнения команды присваивания 2 состояние стека будет $(1 + R1 * \cos(R2), R3)$. Тогда, используя команду $RX = RY$, можно восстановить то состояние RX стека, которое было до выполнения команды 2:

$$3 RX = RY; (3RX = (RX = R3))$$

$$07. F. (R3, R3, R3, 06RX)$$

В общем виде восстановление состояния стека после выполнения одной команды присваивания может быть достигнуто с помощью команды:

Б-команда:
 $n + 0 RY = RX$

П-команда
 $a + 0. F.$

После выполнения двух команд присваивания восстановление первоначального состояния стека может быть достигнуто с помощью команды

Б-команда
 $n \leftarrow 1$ RX = RZ

П-команда
 $a \leftarrow 0$. F.
 $a \leftarrow 1$. F.

2. Использование стековой памяти для хранения значений промежуточных величин. Пусть в регистр X введена некоторая промежуточная величина M, значение которой необходимо сохранить в стековой памяти до окончания выполнения какой-либо вычислительной процедуры, включающей в себя несколько команд присваивания. Это можно сделать с помощью следующей последовательности команд.

Начальное состояние стека ($M, 0, 0, 0$):

A RX = RX; (M, M, M)

$a \leftarrow 0$. B! (M, M)
 $a \leftarrow 1$. B! (M, M, M)

Начальное состояние стека ($M, 0, 0, 0$):

A RT = RX; (M, M, M, M)

$a \leftarrow 0$. B! (M, M)
 $a \leftarrow 1$. B! (M, M, M)
 $a \leftarrow 2$. B! (M, M, M, M)

Значение переменной может быть сохранено в стековой памяти, если поместить его в регистр T (в большинстве случаев достаточно поместить его в регистр Z, поскольку в процессе выполнения первой команды присваивания оно автоматически переместится в регистр T). Когда недостаточно регистров адресуемой памяти, широко используется возможность хранения чисел в регистрах стековой памяти. Однако при этом следует иметь в виду, что количество вводимых переменных, включая и текущие, при вычислении выражений сокращается с трех до двух.

5.2. СОСТАВЛЕНИЕ ЛИНЕЙНЫХ ПРОГРАММ

Алгоритмы и программы будем называть линейными, если их команды выполняются в том порядке, в каком они записаны. Как уже отмечалось в п. 4.1, последовательность операций, выполняемых в линейных алгоритмах и программах, принято называть следованием. В том же параграфе было рассмотрено составление линейного алгоритма и программы расчета площади треугольника (алгоритм 4.0), ориентированных на ручной режим вычислений. Составим теперь алгоритм и программу решения этой задачи, ориентированных на автоматический режим вычислений по предварительно введенной в ГМК программе. Основные различия алгоритма и программы в автоматическом и ручном режимах сводятся к следующему:

1. Алгоритм вычислений в автоматическом режиме трехуровневый (вместо двухуровневого при ручном режиме).

2. На первом уровне представления алгоритма включаются дополнительные команды, предписывающие выполнение ручных операций (ввод исходных данных в вывод результатов; запуск программы и др.).

3. На втором уровне представления дается последовательность только тех команд, автоматизация выполнения которых обеспечивается однозначным их преобразованием в команды программы ГМК. В частности, линейный БЕЙСИК-алгоритм может включать в себя команды только двух типов: команду присваивания и команду STOP.

4. На третьем уровне представления команды НАЖАТЬ заменяются одной последовательностью команд программы, отличающейся от программы ручных вычислений наличием дополнительной команды, предусматривающей останов выполнения программы. Это команда С/П, эквивалентная БЕЙСИК-команде STOP. Состояние стека после выполнения команд будем записывать в основном с помощью внутренних переменных. Для сокращения записи выражений, характеризующих состояние стека, условимся давать ссылки на выражения, записанные в предыдущих командах.

С учетом сделанных замечаний алгоритм и программа вычисления площади треугольника будет выглядеть следующим образом:

ПРОГРАММА 5.1. Вычисление площади треугольника по формуле Герона
 $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a+b+c)/2$.

алг Герон
апт R0 = a, R1 = b, R2 = c
рез RX = S
нач R3 = p
BB R0 = a, R1 = b, R2 = c
НЖ В/О С/П
 $p = (a + b + c)/2$
 $I R3 = (R0 + R1 + R2)/2$
00. пX0 (R0)
01. пX1 (R1, R0)
02. + (R0 + R1)
03. пX2 (R2, R0 + R1)
04. + (R1 + R0 + R2)
05. 2 (2, R0 + R1 + R2)
06. / ((R0 + R1 + R2)/2)
07. хП3 (R3 = p)
 $S = \sqrt{p(p-a)(p-b)(p-c)}$
 $2 \rightarrow RX = SQR(R3 * (R3 - R0) * (R3 - R1) - (R3 * R2))$
08. пX3 (R3, R3)
09. пX0 (R0, R3, R3)
10. - (R3 - R0, R3)
11. × (R3 * (R3 - R0))
12. пX3 (R3, R3 * (R3 - R0))
13. пX1 (R1, R3, 11RX)
14. - (R3 - R1, 11RX)
15. × (11RX * (R3 - R1))
16. пX3 (R3, 15RX)
17. пX2 (R2, R3, 15RX)

18. — (R3 — R2, 15RX)
19. × (15RX * (R3 — R2))
20. F √ (SQR (19RX))
21. C/P (S)

3 STOP (S)

ВЫ S

КОН

ИНСТРУКЦИЯ; ВВ R0=a, R1=b, R2=c; НЖ В/О С/П; ВЫ S
ТЕСТ: ВВ R0=a=0.25; R1=b=0.5; R2=c=0.6; НЖ В/О С/П;
ВЫ S=6.1361505 —02 (10 с)

Как видно из приведенного примера, алгоритм, записанный на БЕЙСИК-ПМК, легко преобразуется в эквивалентную последовательность команд программы. При этом каждая Б-команда интерпретируется соответствующим блоком команд программы. Полученная программа может уже непосредственно восприниматься и выполняться на ПМК. Для этого ее необходимо ввести в память ПМК, предназначенную для хранения программ ОЗУП (оперативное запоминающее устройство программ).

Ввод программы в память ПМК должен производиться до начала вычислений. Перед запуском программы в счетчик адреса команд должен быть занесен пусковой адрес — адрес первой команды программы. Ввод пускового адреса осуществляется путем нажатия клавиш БPnn, где nn — пусковой адрес программы. Если пусковым адресом программы является адрес 00, то установка пускового адреса производится путем нажатия клавиши В/О (возврат обратно).

Запуск программы для вычислений осуществляется путем нажатия клавиши С/С (стоп/пуск). После ее нажатия осуществляется запуск программы, начиная с пускового адреса, занесенного в счетчик адреса команд. По указанному адресу из программной памяти считывается код команды, после чего происходит ее выполнение. Выполнение команды (точнее, шага команды) завершается добавлением единицы в счетчик адреса команд, затем она считывается, выполняется следующая команда и т. д. Последней командой программы, выполняемой в автоматическом режиме, должна быть команда останова вычислений С/П. Ее эквивалентом в БЕЙСИК-алгоритме является команда STOP. Она вводится в память ПМК в составе команд программы путем нажатия той же клавиши С/П, которая использовалась для запуска программы. Таким образом, в ручном режиме нажатие клавиши С/П воспринимается как предписание запуска программы, а в программируемом (в составе программы) — как предписание останова вычислений.

Мы только что рассмотрели линейный алгоритм и линейную программу вычислений. В чистом виде подобные задачи, описываемые линейным алгоритмом и линейной программой, встречаются не так часто. Намного чаще линейные алгоритмы и программы встречаются в качестве составных элементов в алгоритмах с повторениями и ветвлениями. На первом этапе решения таких за-

дач целесообразно автоматизировать выполнение линейных участков алгоритма. При этом условия, включаемые в состав ветвлений и повторений, соблюдаются самим пользователем. На базе алгоритма 4.6, ориентированного на выполнение ручных вычислений, составим алгоритм и программу расчета амплитуды и фазы результирующего колебания, в котором будет предусмотрено автоматическое выполнение на ПМК линейных участков.

ПРОГРАММА 5.2. Расчет амплитуды A и фазы ϕ (в радианах) результирующего колебания, полученного из двух гармонических колебаний с амплитудами A₁ и A₂.

Расчетные формулы:

$$A = \sqrt{1 + n^2 + 2n \cos \psi};$$

$$\psi = \arctg(n \sin \psi / (1 + n \cos \psi)),$$

где n=A₁/A₂; ψ — сдвиг фаз между составляющими гармоническими колебаниями, рад.

```

алг амплитуда — фаза
арг таб R0=ψ; R1=n
рез таб RX=A, RX=ϕ

нач i, j, RX= q
РАДИАНЫ
для i от 1 до длин (A)
нц BBR0 = ψ [i]
для j от 1 до шир (A)
нц
    ВВ R1=n |j|
    A = √1 + n (j) ** 2 + 2 * n |j| * cos ψ [i]
    1 RX = SQR (1 + R1 * * 2 + 2 * R1 * COS (R0))
    00. 1 (1)
    01. πX1 (R1, 1)
    02. Fx2 (R1 * * 2, 1)
    03. + (1 + R1 * * 2)
    04. 2 (2, 1 + R1 * * 2)
    05. πX1 (R1, 2, 1 + R1 * * 2)
    06. × (2 * R1, 1 + R1 * * 2)
    07. πX0 (R0, 2 * R1, 1 + R1 * * 2)
    08. F cos (COS (R0), 2 * R1, 03RX)
    09. × (2 * R1 * COS (R0), 03RX)
    10. + (03RX + 09RX)
    11. F √ (SQR (10RX))
    12. C/P (A)

    2 STOP (A)
    ВЫ A
    q = 1 + n |j| * cos ψ [i]
    3 RX = 1 + R1 * COS (R0)
    13. πX1 (R1)
    14. πX0 (R0, R1)
    15. F cos (COS (R0), R1)
    16. × (R1 * COS (R0))
    17. 1 (1, R1 * COS (R0))
    18. + (1 + R1 * COS (R0))
    19. C/P (q)

    4 STOP
    ВЫ q

```

```

если  $q \neq 0$ ,  

то  $\varphi[i, j] = \operatorname{arctg}(n[j] * \sin \psi[i]/q)$   

5 RX = ATG(R1 * SIN(R0)/RX)
20. nX1(R1, q)
21. nX0(R0, R1, q)
22. F sin(SIN(R0), R1, q)
23. X(R1 * SIN(R0), q)
24. ↔(q, R1 * SIN(R0))
25. ÷(R1 * SIN(R0)/q)
26. F tg⁻¹(φ)
27. C/P(φ)

```

6 STOP
ВЫ φ[i, j]
иначе ВЫ φ[i, j] = π/2

все

кц

КОН
ИНСТРУКЦИЯ
РАДИАНЫ
для i от 1 до длин (A)

нц

ВВ R0 = ψ[i]
для j от 1 до шир (A)

нц
ВВ R1 = n[j]; НЖ В/О С/П
ВЫ A[i, j]; НЖ С/П
ВЫ q
если $q = 0$,
то НЖ В/О С/П; ВЫ φ[i, j]
иначе ВЫ φ[i, j] = π/2

все

кц

ТЕСТ

ВВ R0 = ψ = 3π/2, R1 = n = 2; НЖ В/О С/П
ВЫ A = 2.2360679 (3 с); НЖ С/П
ВЫ $1 + n \cos \psi = 1$ (3 с); НЖ С/П
ВЫ φ = -1.1071487 (5 с)

Здесь программа для ПМК является линейной, поскольку она выполняется строго в порядке возрастания адресов ее команд. Однако задача и алгоритм ее решения в целом являются нелинейными, поскольку они включают в себя повторения и ветвления. В рассматриваемой задаче автоматизируется лишь часть общего вычислительного процесса (его линейная часть). Другая его часть (нелинейная) никак не отражается в программе для ПМК и выполняется пользователем вручную.

Упражнения

5.2.1. Разработать алгоритм и программу расчета значений надежности P системы, состоящей из n одинаковых параллельно соединенных элементов, каждый из которых обладает надежностью p . Расчет произвести по формуле

$$P = (1 - (1 - p)^n).$$

5.2.2. Разработать алгоритм и программу расчета на ПМК таблицы функций, определяемой формулой

$$Y = (A + Bx)/((A + x)(B + x)).$$

5.2.3. В ПМК типа «Электроника» не предусмотрены команды вычисления гиперболических функций. Составить алгоритмы и программы вычисления гиперболических функций, используя формулы:

- 1) $\operatorname{sh} x = (e^x - e^{-x})/2$;
- 2) $\operatorname{ch} x = (e^x + e^{-x})/2$;
- 3) $\operatorname{th} x = (e^x - e^{-x})(e^x + e^{-x})$.

5.3. ВВОД ПРОГРАММЫ В ПАМЯТЬ ПМК

Для ввода программы необходимо:

1. Ввести в счетчик адреса команд (СЧАК) пусковой адрес программы, нажав клавиши БПп или В/О.
2. Перевести ПМК из режима вычислений в режим ввода программы, нажав клавиши ГПРГ.
3. Набрать на пульте ПМК текст программы.
4. Перевести ПМК из режима ввода программы в режим вычислений, нажав клавиши ФАВТ.

Назначение всех клавиш, используемых в процессе ввода, отладки и запуска программы на выполнение, приведено в табл. 5.1.

Таблица 5.1

Команды (клавиши), используемые для ввода, отладки программы и управления решением задач в программируемом режиме

Команда (клавиша)	Назначение
В/О	Сброс счетчика адреса команд. При нажатии В/О в него заносится адрес 00. Используется для занесения пускового адреса программы, выполнение которой должно начинаться с адреса 00
БПп	Безусловный переход к выполнению команды, адрес которой указан за символами БП. Адрес ячейки программной памяти ПМК, который предписывается занести в счетчик адреса команд. Если $nn > 99$, то вместо первого n в команде указывается точка, например, вместо $nn = 103$ в команде указывается адрес .3 Используется для: занесения пускового адреса программы, отличающегося от адреса 00; установки адреса команды программы, подлежащей исправлению
С/П	1. Запуск программы, записанной в программной памяти ПМК с адреса, хранящегося в счетчике адреса команд, если программа не находится в стадии выполнения. 2. Останов вычислений, если программа уже запущена в работу и находится в стадии ее выполнения. Выполняется, если: команда (С/П) встречается в любом месте текста программы; после нажатия клавиши С/П вручную в процессе выполнения программы

Команда (клавиша)	Назначение
FПРГ	Перевод ПМК из режима вычислений в режим ввода программы. Выполнение команд ПМК блокируется (кроме ШГ и ШГ). После нажатия клавиш на пульте управления производится запись в программную память кодов соответствующих клавиш (команд). Запись начинается с адреса, сохранившегося в СЧАК в момент перевода ПМК из режима вычислений в режим ввода программы
FАВТ	Обратный перевод ПМК из режима ввода программы в режим вычислений с сохранением содержимого счетчика адреса команд (СЧАК). После перехода в режим вычислений в СЧАК будет храниться адрес, на единицу больший адреса последней команды программы. Блокировка выполнения команд снимается, ПМК готов к выполнению вычислений как в обычном, так и в программируемом режиме. При выполнении вычислений в обычном режиме программа, введенная в программную память ПМК, сохраняется
ШГ	Просмотр содержимого программной памяти в режиме ввода программы. После нажатия клавиши содержимое СЧАК уменьшается на единицу. На экране справа отображается новое содержимое СЧАК. На индикаторе слева отображается код команды, адрес которой на единицу меньше содержимого СЧАК
ШГ	То же, что и ШГ, только при этом содержимое СЧАК не уменьшается, а увеличивается на единицу
ПП	Пошаговое (покомандное) выполнение программы. Используется при отладке программы в режиме вычислений. При нажатии клавиши ПП выполняется очередная команда программы, записанная в памяти ПМК (т. е. та, адрес которой находится в СЧАК). После ее выполнения содержимое СЧАК увеличивается на единицу. Для выполнения следующей команды программы необходимо снова нажать на клавиши ПП
Предупреждение. 1. Клавиши В/О, БПнп, С/Г и ПП следует нажимать, когда ПМК находится в режиме вычислений. Попытка нажатия этих клавиш в режиме ввода программы (после нажатия клавиши FПРГ) приведет к тому, что это будет воспринято как ввод в программную память соответствующих команд, и к появлению ошибок в программе. 2. При выключении питания все регистры ПМК (в том числе и ячейки программной памяти) обнуляются. Поэтому для сохранения программы в программной памяти ПМК нельзя отключать питание. В противном случае программу придется вводить заново.	

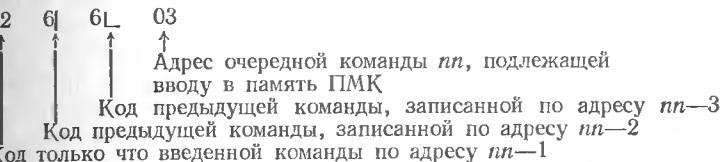
Рассмотрим последовательность ввода программы более подробно:

1. Ввод в счетчик адреса команд пускового адреса программы. Производится в режиме выполнения вычислений (перевод ПМК из режима ввода программы в режим вычислений выполняется путем нажатия клавиш FАВТ). Осуществляется путем нажатия клавиш БПнп, где nn — пусковой адрес программы (или В/О, когда nn = 00).

2. Перевод ПМК из режима вычислений в режим ввода программы. Осуществляется путем нажатия клавиш FПРГ. После нажатия этих клавиш на экране индикатора (в правой части) будет отображен только что введенный пусковой адрес программы (если для его ввода использовалась клавиша В/О, то на экране справа будет отображено 00).

3. Ввод текста программы с клавиатуры ПМК. Производится последовательно в порядке номеров (адресов) команд, начиная с пускового и кончая последним. При этом вводу подлежат только сами команды. Адреса команд не вводятся. Они устанавливаются автоматически с помощью счетчика адреса команд ПМК, причем каждый раз в правой части индикатора отображается адрес, по которому будет занесена в программную память следующая команда.

После ввода каждой команды ПМК на экране индикатора отображается последовательность из четырех групп символов:



Таким образом, на экране индикатора отображаются не обозначения команд, соответствующие их надписям на клавиши, а их коды, в которых команды записываются в ячейках программной памяти после нажатия соответствующих клавиш.

Как видно из приведенного примера, адрес команды на экране индикатора всегда отображается двумя символами. В МК-54 адресное поле ограничено адресами 00–97, поэтому для отображения каждого из них на экране достаточно двух десятичных цифр. В МК-52 и МК-61 адресное поле памяти шире, чем в МК-54. Поэтому двумя десятичными цифрами отображаются адреса, начиная с 00 и кончая 99. Отображение последующих адресов осуществляется также с помощью двух символов, первый из которых знак минус, а второй — десятичная цифра, т. е. на экране индикатора последующие адреса будут отображаться так:

98, 99, -0, -1, -2, -3, -4.

При вводе программы вычисления амплитуды A (пусковой адрес равен нулю) после ввода каждой команды на экране индикатора будут последовательно отображаться следующие символы:

Адреса команд	Обозначения клавиш	Отображение на экране индикатора после ввода команды			
—	—	—	—	—	00
00.	1	01	—	—	01
01.	пХ1	61	01	—	02
02.	Fx ²	22	61	01	03
03.	+	10	22	61	04
04.	2	02	10	22	05
05.	пХ1	61	02	10	06

06.	X	12	61	02	07
07.	nX0	60	12	61	08
08.	F cos	0	60	12	09
09.	X	12	11	60	10
10.	+	10	12	IГ	11
11.	F V	21	10	12	12
12.	C/P	50	21	10	13

Прежде чем начать ввод очередной команды необходимо:
сверить код только что введенной команды (отображаемый слева на экране индикатора) с обозначением команды в тексте программы;

сверить адрес очередной команды, отображаемой на экране индикатора справа с номером команды в тексте программы, которую мы собираемся вводить. И лишь при их совпадении можно нажать на клавишу ввода очередной команды. Таким образом, вводится вся программа до последней команды. Контроль правильности ввода команд программы можно выполнить с помощью таблицы перехода коды команд — обозначения команд (табл. 5.2). В этой таблице обозначение каждой команды находится на пересечении строки и столбца, соответствующих шестнадцатиричным цифрам ее кода, отображаемого на индикаторе. При этом первая цифра кода соответствует шестнадцатиричному номеру строки, а вторая — столбца. Кодовые комбинации, не используемые в ПМК, обозначены в таблице звездочками.

Если при вводе программы была обнаружена ошибка (например, нажата не та клавиша), то любую команду, записанную в программную память ПМК, можно исправить, записав на ее место

другую (правильную) команду. Для этого достаточно в счетчике адреса команд установить адрес команды, подлежащей исправлению (при установке СЧАК требуемого адреса они будут отображаться в правой части экрана индикатора). Изменение состояния этого счетчика осуществляется путем нажатия клавиш $\overleftarrow{ШГ}$

(содержимое СЧАК при нажатии уменьшается на единицу) и $\overrightarrow{ШГ}$ (содержимое СЧАК при нажатии увеличивается на единицу).

Установив с помощью клавиш $\overleftarrow{ШГ}$ и $\overrightarrow{ШГ}$ адрес команды, которую необходимо исправить можно нажать клавишу, соответствующую правильной команде. Она будет введена в ячейку программной памяти по указанному адресу. Старая (неправильная) команда при этом сотрется.

Как показывает практика, при вводе программы в большинстве случаев нет необходимости следить за правильностью воспроизведения кодов команд на экране индикатора. Достаточно следить за совпадением адресов в тексте программы и на экране индикатора. При несовпадении ввод повторяется с того места программы, где есть уверенность что все предыдущие команды введены правильно.

4. Перевод ПМК из режима ввода программы в режим вычислений. Осуществляется с помощью команды FABT. После выполнения этой команды на ПМК могут производиться вычисления как в обычном, так и программируемом режимах, причем вычисления, выполняемые в обычном режиме, никак не влияют на сохранность программы, введенной в программную память ПМК.

Перевод кодов команд в обозначения

Коды	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	+	-	X	\div	\leftrightarrow	$F10^x$	Fe^x	Flg
2	Fπ	F V	Fx^2	$F1/x$	Fy	*	$K_0^>$	*
3	$K_0^{<...}$	$K x $	K3H	$K_0^{>}$	$K[x]$	$K\{x\}$	K_{max}	$K\wedge$
4	$x10$	$x\Pi$	$x\Pi2$	$x\Pi3$	$x\Pi4$	$x\Pi5$	$x\Pi6$	$x\Pi7$
5	C/P	B/P	B/O	P/P	KНОП	*	*	$Fx = 0$
6	nX0	PX1	PX2	PX3	PX4	PX5	PX6	PX7
7	$Kx \neq_0$	$Kx \neq_1$	$Kx \neq_2$	$Kx \neq_3$	$Kx \neq_4$	$Kx \neq_5$	$Kx \neq_6$	$Kx \neq_7$
8	KБП0	KБП1	KБП2	KБП3	KБП4	KБП5	KБП6	KБП7
9	$Kx \geqslant_0$	$Kx \geqslant_1$	$Kx \geqslant_2$	$Kx \geqslant_3$	$Kx \geqslant_4$	$Kx \geqslant_5$	$Kx \geqslant_6$	$Kx \geqslant_7$
L	KПП0	KПП1	KПП2	KПП3	KПП4	KПП5	KПП6	KПП7
I	$Kx <_0$	$Kx <_1$	$Kx <_2$	$Kx <_3$	$Kx <_4$	$Kx <_5$	$Kx <_6$	$Kx <_7$
G	KпХ0	KпХ1	KпХ2	KпХ3	KпХ4	KпХ5	KпХ6	KпХ7
E	$Kx =_0$	$Kx =_1$	$Kx =_2$	$Kx =_3$	$Kx =_4$	$Kx =_5$	$Kx =_6$	$Kx =_7$
	*	*	*	*	*	*	*	*

Таблица 5.2
клавиш на клавиатуре ПМК

8	9	-	L	I	Г	E	-
8	9						
Fln	$F \sin^{-1}$	$F \cos^{-1}$	$F \tg^{-1}$	$F \sin$	$F \cos$	$F \tg$	*
*	*	$K_0^{>...}$	*	*	*	*	*
KV	$K\oplus$	KИНВ	KСЧ	*	*	*	*
$x\Pi8$	$x\Pi9$	$x\Pi A$	$x\Pi B$	$x\Pi C$	$x\Pi D$	$x\Pi E$	*
FL2	$Fx \geqslant 0$	FL3	$Fx < 0$	FL0	$Fx = 0$	*	*
nX8	nX9	pXA	pXB	nXC	nXD	pXE	*
$Kx \neq_0 8$	$Kx \neq_0 9$	$Kx \neq_0 A$	$Kx \neq_0 B$	$Kx \neq_0 C$	$Kx \neq_0 D$	$Kx \neq_0 E$	*
KБП8	KБП9	KБПA	KБПB	KБПC	KБПD	KБПE	*
$Kx \geqslant_0 8$	$Kx \geqslant_0 9$	$Kx \geqslant_0 A$	$Kx \geqslant_0 B$	$Kx \geqslant_0 C$	$Kx \geqslant_0 D$	$Kx \geqslant_0 E$	*
KПП8	KПП9	KППA	KППB	KППC	KППD	KППE	*
KxП8	KxП9	KxПA	KxПB	KxПC	KxПD	KxПE	*
$Kx <_0 8$	$Kx <_0 9$	$Kx <_0 A$	$Kx <_0 B$	$Kx <_0 C$	$Kx <_0 D$	$Kx <_0 E$	*
KпX8	KпX9	KпXA	KпXB	KпXC	KпXD	KпXE	*
$Kx =_0 8$	$Kx =_0 9$	$Kx =_0 A$	$Kx =_0 B$	$Kx =_0 C$	$Kx =_0 D$	$Kx =_0 E$	*

5.4. ОТЛАДКА ПРОГРАММЫ

При разработке программы, записи ее на бумаге, вводе в программную память очень часто возникают ошибки. Поэтому после разработки программы необходим этап ее отладки. Отладка программы — это процесс обнаружения и исправления ошибок в ней, а также установления факта правильного ее функционирования, что существует с помощью контрольного решения. Отладка предполагает сравнение промежуточных и окончательных результатов с некоторыми эталонными (например, полученными на ПМК в обычном режиме его использования). Назначение клавиш ПМК, используемых в процессе отладки программы, дано в табл. 5.1.

Процесс отладки программы включает в себя следующие этапы.

1. Получение эталонного решения задачи.
2. Выполнение программы в отладочном режиме и обнаружение ошибок.
3. Исправление ошибок в программе, записанной в памяти ПМК.
4. Выполнение контрольного решения по отлаженной программе.

Получение эталонного решения. Для получения эталонного решения чаще всего используется ПМК в обычном режиме вычислений. Исходные данные для эталонного решения должны быть выбраны такие, чтобы после их ввода и проведения требуемых расчетов обеспечивалась бы проверка выполнения всех ветвей, блоков и команд программы. Если в задаче расчета амплитуды A и фазы φ результирующего колебания в качестве исходных данных взять $\psi = 3\pi/2$ и $n = 2$, то $1 + n \cos \psi \neq 0$. Это позволит проверить все три части программы, вычисляющие соответственно A , $1 + n \cos \psi$ и φ . Если же взять $\psi = \pi/2$ и $n = 1$, то проверить команды вычисления φ не удастся, так как при этом $1 + n \cos \psi = 0$.

Может оказаться, что одного варианта исходных данных, с помощью которого были бы проверены все команды программы, подобрать не удастся. Тогда подбирают несколько таких вариантов, чтобы, последовательно решив задачу для всех подобранных вариантов, можно было проверить все команды программы.

В качестве исходных данных для эталонного решения выберем значение $\psi = 3\pi/2$ и $n = 2$. Как будет видно в дальнейшем, при таких значениях исходных данных будут задействованы (а следовательно, и могут быть проверены) все команды программы. Этапонное решение обычно выполняется в ручном режиме, как это было показано в алгоритме 4.6. Только в отличие от него после каждой команды будем перечислять в стеке не только имена переменных, но и указывать их конкретные значения, отображаемые на экране индикатора. Для выбранных значений ψ и n эталонное решение можно записать следующим образом:

РАДИАНЫ
ВВЕСТИ R0 = $\psi = 3\pi/2$, R1 = $n = 2$

НАЖАТЬ

00. 1 (I)
01. $\pi X1 (n = 2, 1)$
02. $Fx^2 (n^2 = 4, 1)$
03. $+ (n^2 + 1 = 5)$
04. $2 (2, 1 + n^2)$
05. $\pi X1 (2, 2, 1 + n^2)$
06. $\times (2n = 4, 1 + n^2)$
07. $\pi X0 (\psi = 3\pi/2 = 4.7123889, 2n, 1 + n^2)$
08. $F \cos (\cos \psi = 0, 2n, 1 + n^2)$
09. $\times (2n \cos \psi = 0, 1 + n^2)$

$$10. + (1 + n^2 + 2n \cos \psi = 5)$$

$$11. F \sqrt{A = \sqrt{1 + n^2 + 2n \cos \psi}} = 2.2360679$$

ВЫДАТЬ $A = 2.2360679$

НАЖАТЬ

12. $\pi X1 (n = 2)$
13. $\pi X0 (\psi = 3\pi/2 = 4.7123889, n)$
14. $F \cos (\cos \psi = 0, n)$
15. $\times (n \cos \psi)$
16. $1 (1, n \cos \psi)$
17. $+ (1 + n \cos \psi = 1)$

если $1 + n \cos \psi \neq 0$

то НАЖАТЬ

18. $\pi X1 (n = 2, 1 + n \cos \psi)$
19. $\pi X0 (\psi = 3\pi/2 = 4.7123889, n, 1 + n \cos \psi)$
20. $F \sin (\sin \psi = -1, n, 1 + n \cos \psi)$
21. $\times (n \sin \psi = -2, 1 + n \cos \psi)$
22. $\leftrightarrow (1 + n \cos \psi = 1, n \sin \psi)$
23. $\div ((n \sin \psi)/(1 + n \cos \psi) = -2)$
24. $F \operatorname{tg}^{-1} (\varphi = \operatorname{arctg}((n \sin \psi)/(1 + n \cos \psi)))$

ВЫДАТЬ $\varphi = -1.1071487$

иначе ВЫДАТЬ $\varphi = \pi/2$

все

Выполнение программы в отладочном режиме. Различие выполнения программы в автоматическом и отладочном режимах состоит в том, что в последнем она производится по командам, причем после завершения каждой команды происходит останов вычислений и на экране индикатора отображается результат. В отладочном режиме программа предварительно должна быть записана в программную память ПМК. После нажатия клавиши ПП выполняется одна очередная команда программы и вслед за этим происходит останов вычислений. Во время останова необходимо просмотреть на экране индикатора и убедиться, что полученный результат совпадает с эталонным. В противном случае в последней выполненной команде имеется ошибка. Необходимо определить адрес этой команды и ее код (содержимое ячейки памяти, в которой она хранится) и исправить ее. Как это делается, будет показано ниже.

Для определения адреса только что выполненной команды достаточно перевести ПМК в режим ввода программы и прочитать в правой части индикатора содержимое счетчика адреса команд. Оно будет на единицу больше, чем адрес только что выполненной команды. При этом код выполненной команды будет отображен в левой части индикатора.

Исправив неправильно введенную команду, необходимо заново запустить программу, начиная с адреса исправленной команды, и так далее до тех пор, пока не будут исправлены все ошибки в программе и не получен правильный результат окончательного решения, совпадающего с эталонным. После этого, нажав клавиши В/О и С/П (или БПн, если пусковой адрес программы не равен нулю), необходимо проверить работу программы в автоматическом режиме и установить время контрольного решения.

Прежде всего. Проверку работы программы в автоматическом режиме следует производить обязательно, так как по опыту эксплуатации ПМК замечено, что программа, проверенная в отладочном режиме (путем последовательного нажатия клавиши ПП), не всегда правильно работает в автоматическом режиме (что часто бывает при использовании в программе команды В!).

Исправление ошибок в введенной программе. В процессе исправления ошибок в программе чаще всего выполняются следующие операции:

вставляется команда в программу (между двумя другими командами — операция ВСТАВИТЬ);

удаляется команда из программы (операция УДАЛИТЬ); заменяется команда в программе (операция ЗАМЕНИТЬ).

Рассмотрим на примерах, как выполняются эти операции.

1. Операция ВСТАВИТЬ. Пусть в программе оказалась пропущенной какая-то команда, например 09.×. Для того чтобы вставить эту команду между командами 08. Fcos и 09. +, необходимо:

1) Если пропуск команды обнаружен в режиме ввода программы, то с помощью последовательного нажатия клавиш $\overleftarrow{ШГ}$ и (или) $\overrightarrow{ШГ}$ установить показания счетчика адреса команд на индикаторе справа, соответствующие адресу пропущенной команды 09. На экране индикатора будет в рассматриваемом случае отображено 1Г 60 12 09. После этого необходимо ввести пропущенную команду 09.×, затем 10.+ и все остальные команды до конца программы.

2) Если пропуск команды был обнаружен в режиме вычислений, то необходимо набрать на пульте БПн, где nn — адрес команды, которую необходимо вставить (в данном случае БП 09), а затем перевести ПМК в режим ввода программы, нажав клавиши ГПРГ, после чего на экране индикатора будут отображен адрес вставляемой команды и коды всех предшествующих команд: 1Г 60 12 09. Затем необходимо ввести вставляемую команду и

все последующие команды в память ПМК вплоть до последней команды программы. После ввода вставляемой команды показания индикатора в нашем случае будут такими: 10 12 1Г 11.

2. Операция ЗАМЕНИТЬ. Пусть оказалось, что в команде 10.+ ошибочно оказался знак умножения, т. е. вместо данной команды программной памятью оказалась записанной команда 10.×. Требуется заменить эту команду на правильную. Если необходимость замены обнаружится, когда ПМК находится в режиме ввода программы, то с помощью клавиш $\overleftarrow{ШГ}$ и/или $\overrightarrow{ШГ}$ на экране индикатора устанавливаются показания счетчика адреса команд, соответствующие адресу заменяемой команды. Если же необходимость замены обнаружилась, когда ПМК находился в режиме вычислений, то следует нажать на клавиши БПн, где nn — адрес заменяемой команды (в данном случае 10.+), а затем перевести ПМК в режим ввода программы. Тогда в счетчике адреса команд будет установлен адрес заменяемой команды и на экране индикатора в обоих случаях будет отображено 12 1Г 60 10, где 10 — адрес команды, которую необходимо заменить, 12 — код предыдущей команды программы. После этого необходимо ввести в память ПМК заменяемую команду 10.+, нажав клавишу +. На экране индикатора будет отражено 10 12 1Г 11, где 11 — адрес следующей команды, а 10 — код только что замененной команды сложения. Старая, неправильно введенная команда при этом стирается.

3. Операция УДАЛИТЬ. Пусть имеется последовательность команд программы:

00. nX0 (R0)

01. nX1 (R1, R0)

02. nX2 (R2, R1, R3)

03. + (R1+R2, R0)

04. nX0 (R0, R1+R2, R0)

05. × (R0*R1+R2, R0)

Нетрудно заметить, что команда 04. nX0 является лишней, поскольку значение содержимого регистра 0 еще находится и в регистре Y и можно сразу выполнить операцию 05.×. Тогда число команд программы можно сократить на одну, исключив команду 04. (nX0 обведена в рамку).

Операцию УДАЛИТЬ можно было бы выполнить путем пересадки всех команд, начиная с команды, непосредственно следующей за удаляемой. При этом адреса всех переадресуемых команд уменьшаются на единицу. В частности, 05.× становится командой 04.×, а старая команда 0.4 nX0 стирается. Однако при отладке программ команду, подлежащую исключению, чаще всего забывают, помещая по ее адресу пустую команду КНОП (нет операции), но учитываемую счетчиком адреса команд как обычную команду. Если вместо удаляемой команды мы поместим команду КНОП, то адреса всех последующих команд останутся неизменными. В последнем случае операция УДАЛИТЬ выполняется точно так же, как и операция ЗАМЕНИТЬ. При этом в качестве заменяющей команды используется команда КНОП.

Рекомендации по предупреждению возможных ошибок в программе.

1. При написании текста программы обязательно следует записывать адреса команд и состояние всех стековых регистров после выполнения команды.

2. При вводе программы необходимо тщательно следить за совпадением адреса, отображаемого на экране индикатора, с адресом текущей вводимой команды.

Выполнение контрольного решения в автоматическом режиме. Если в результате работы программы в отладочном режиме будет получено совпадение результатов выполнения каждой ее команды с эталонными результатами, то можно приступить к выполнению контрольного решения по введенной программе в автоматическом режиме по эталонным исходным данным. Цель такого решения — установить:

факт нормального функционирования отложенной программы; ориентировочное время выполнения программы.

Для выполнения контрольного решения в автоматическом режиме необходимо:

1. Перевести ПМК в режим вычислений (нажать клавиши FABT).

2. Ввести в ПМК эталонные исходные данные, необходимые для выполнения программы.

3. Ввести в ПМК пусковой адрес программы, нажав клавиши БПп (или В/О при $nn = 00$).

4. Запустить программу на автоматическое выполнение, зафиксировав время запуска (нажатия клавиши С/П).

5. Зафиксировав время появления результата на экране индикатора, прочесть его и сопоставить с соответствующим результатом эталонного решения. Их совпадение устанавливает факт правильного функционирования программы, и процесс отладки заканчивается установлением ориентировочного времени решения задачи по отложенной программе.

Знание времени контрольного решения необходимо, чтобы следить за правильностью выполнения программы. Если окажется, что время выполнения программы намного превышает время контрольного решения, то это будет означать, что программа работает неправильно. Ее следует остановить, нажав клавишу С/П, а затем, проверив правильность исходных данных в регистрах памяти ПМК и содержимого СЧАК, вновь запустить на автоматическое выполнение.

Контрольное решение играет роль испытательного теста для разработанной и отложенной программы. Тест, начинающийся с заголовка ТЕСТ, за которым следуют варианты эталонного решения, является неотъемлемой принадлежностью любой программы и всегда должен записываться после окончания ее основного текста. Примеры записи испытательных тестов были приведены ранее (в программах 5.1 и 5.2).

5.4.1. ЗАПИСЬ ОТЛАЖЕННОЙ ПРОГРАММЫ В ППЗУ (ТОЛЬКО ДЛЯ МК-52)

Размещение программ в ППЗУ. Во всех ПМК, выпускаемых нашей промышленностью, при выключении питания стирается программа, хранящаяся в программной памяти. В МК-52 имеется специальная энергонезависимая память ППЗУ емкостью в 512 байт, позволяющая хранить в ней программы и данные в течение до 5000 ч при отключенном электропитании (см. п. 1.2.2).

Для записи отложенной программы в ППЗУ необходимо прежде всего знать, куда ее можно поместить, не повредив другие программы, т. е. необходимо определить поле памяти ППЗУ, предназначенное для хранения записываемой программы. Для этого пользователю МК-52 необходимо ввести учет распределения памяти ППЗУ, с помощью которого всегда можно было бы узнать, какие программы записаны в ППЗУ, какое поле памяти занимает каждая программа, каков ее начальный адрес в ППЗУ, какова ее длина в байтах, какая команда должна быть набрана на пульте ПМК для обращения к каждой записанной в ППЗУ программе.

ППЗУ в МК-52 устроено таким образом, что обмен программами и данными между ОЗУП и ППЗУ в режимах записи и считывания осуществляется пакетами по 7 байт, а их стирание в ППЗУ — пакетами по 8 байт *. Поэтому, чтобы исключить повреждение соседних программ при записи и стирании, поле ППЗУ, предназначенное для хранения программы, должно быть по сравнению с ее длиной в шагах увеличено до количества байт, кратным 7 (для записи и считывания) и 8 (для стирания). Тогда длина поля памяти ППЗУ в байтах D_b , которое необходимо выделить для хранения программы длиной в D_k команд (точнее шагов), может быть определена с помощью алгоритма:

алг Длина поля ППЗУ

арг D_k

рез D_b

нач D

если $F(D_k/7) \neq 0$ * F — дробная часть числа *

то $D = (E(D_k/7) + 1) * 7$ * E — целая часть числа *

иначе $D = 7 * E(D_k/7)$

все

если $F(D_b/8) \neq 0$

то $D_b = (E(D_b/8) + 1) * 8$

иначе $D_b = 8 * E(D_b/8)$

все

кон

* Подробнее см. «Наука и жизнь», 1989, № 1, с. 124—129.

Ранее были разработаны (упражнения 5.2.3) программы вычисления гиперболических функций $\text{sh } x$ (программа 1, 9 команд), $\text{ch } x$ (программа 2, 9 команд) и $\text{th } x$ (программа 3, 14 команд). Для их размещения в ППЗУ согласно приведенному алгоритму потребуется память Дб (в байтах):

$$\begin{array}{llllllll} \text{sh } x & 9/7 = 1.28 & (1+1)*7 = 14 & 14/8 = 1.75 & (1+1)*8 = 16 \text{ байт} \\ \text{ch } x & 9/7 = 1.28 & (1+1)*7 = 14 & 14/8 = 1.75 & (1+1)*8 = 16 \text{ байт} \\ \text{th } x & 14/7 = 2.0 & 2*7 = 14 & 14/8 = 1.75 & (1+1)*8 = 16 \text{ байт} \end{array}$$

Программы рекомендуется записывать и хранить впритык друг к другу. Тогда адрес начального байта каждой следующей $(n+1)$ -й программы будет определяться так:

$$\text{Ab}(n+1) = \text{Ab}(n) + \text{Db}(n),$$

где $\text{Ab}(n)$ — адрес начального байта предыдущей программы; $\text{Db}(n)$ — требуемая память для ее размещения в ППЗУ, байт.

Заполнение ППЗУ программами обычно начинают с нулевого байта, поэтому адрес начального байта первой программы выберем равным $\text{Ab}(n) = 000$. Тогда адрес начального байта второй программы ($\text{ch } x$)

$$\text{Ab}(2) = \text{Ab}(1) + \text{Db}(1) = 000 + 16 = 016,$$

а третьей ($\text{th } x$)

$$\text{Ab}(3) = \text{Ab}(2) + \text{Db}(2) = 016 + 16 = 032.$$

Адрес начального байта четвертой программы, которую еще предстоит записать в ППЗУ,

$$\text{Ab}(4) = \text{Ab}(3) + \text{Db}(3) = 032 + 16 = 048.$$

Учет размещения в ППЗУ программ ведется с помощью табл. 5.3.

Команда обращения к ППЗУ имеет следующую структуру:

1003216
 ↑↑
 Число байт ППЗУ, необходимое для размещения программы Дб
 Адрес начального символа программы в ППЗУ полубайта Ас ($\text{Ac} = 2 \text{ Ab}$)
 Признак команды обращения к ППЗУ (может быть любой цифрой, кроме нуля)

В команде обращения к ППЗУ указывается адрес не начального байта программы, а начального ее полубайта. В полубайт помещается один символ — одна шестнадцатеричная цифра. Код команды в ОЗУП и ППЗУ состоит из двух таких цифр. Адрес начального полубайта определяется путем удвоения адреса начального байта и записи первого в графе 4 табл. 5.3 в виде четырех цифр.

Распределение памяти ППЗУ

Название программы	Длина программы ППЗУ, Дб, байт	Адрес начального байта в ППЗУ Аб	Адрес начального символа в ППЗУ Ас	Команда обращения к ППЗУ
sh x	16	000	0000	1000016
ch x	16	016	0032	1003216
th x	16	032	0064	1006416
A, φ	32	048	0096	1009632
...	...	080	0160	10160...

Принятый формат команды обращения к ППЗУ ограничивает длину хранимой там программы максимальным двузначным числом, т. е. 99 байтами, а с учетом требования кратности семи — 98 байтами. В то же время емкость ОЗУП составляет 105 байт (для МК-54 — 98 байт), т. е. вводимая вручную в ОЗУП программа может иметь несколько большую длину, но тогда она не сможет целиком быть записанной в ППЗУ.

Очистка поля ППЗУ, на которое должна быть записана отложенная программа. Если поле ППЗУ, на которое должна быть записана очередная программа, использовалось ранее для хранения других программ (числовых данных), то их необходимо стереть. Иначе вновь записанная на это место программа может быть искажена. Для гарантии ее сохранности место, в которое она должна быть записана, всегда должно быть предварительно очищено. Очистка ОЗУ от старой информации должна производиться заранее, еще до того, как программа, которую нужно записать, будет введена в ОЗУП. Дело в том, что при стирании старой информации в ППЗУ одновременно стирается и информация в ОЗУП и, естественно, после этого переписывать из ОЗУП в ППЗУ уже будет нечего.

Для очистки поля ППЗУ, предназначенного для записи новой программы, необходимо выполнить следующие действия:

1. Перевести переключатель Д—П в положение П (программы), а переключатель С—З—СЧ—в положение С (стирание).

2. Определить с помощью табл. 5.3 команду обращения к ППЗУ. Положим, что в ППЗУ хранятся уже три программы. Их необходимо сохранить и очистить поле для четвертой, длина которой берется максимальной (98 байт). Тогда команда обращения к ППЗУ в режиме стирания будет следующей:

1009698
 ↑↑
 Длина поля, подлежащего стиранию в байтах
 Адрес начального полубайта поля программы в ППЗУ
 Признак команды обращения к ППЗУ

3. Набрать на клавиатуре адрес обращения к ППЗУ (1009698).

4. Нажать клавишу $A \uparrow$, предписывающую занести команду обращения из регистра Х в регистр команд ППЗУ и запомнить его. Выполнение команды А сопровождается кратковременным появлением во всех разрядах индикатора знаков минус при сохранении команды обращения к ППЗУ на экране индикатора.

5. Нажать клавишу $\uparrow \downarrow$ (обмен между ОЗУ и ППЗУ), в результате чего произойдет стирание информации в поле ППЗУ, границы которого заданы командой обращения к ППЗУ. В случае команды 1009698 будет очищено 104 байта памяти ППЗУ (кратное 8), начиная с 48-го байта. Одновременно будет очищено и 98 байт ОЗУП. Выполнение команды также сопровождается кратковременным появлением во всех разрядах индикатора знаков минус на фоне цифр команды.

Запись отложенной программы в ППЗУ. Пусть требуется записать в ППЗУ отложенную программу расчета амплитуды и фазы результирующего колебания (программа A и ф) в табл. 5.3, которая уже находится в ОЗУП. Эта программа состоит из 28 команд. Для ее размещения в ППЗУ требуется $D_b = 32$ байта ($28/7 = 4$; $28/8 = 3.5$; $(3 + 1) * 8 = 32$). Чтобы записать программу, находящуюся в ОЗУП, необходимо выполнить следующие действия.

1. Перевести ПМК в режим вычислений, нажав клавиши FABT.

2. Определить по табл. 5.3 команду обращения к ППЗУ для очередной записываемой программы. Для программы расчета A и ф длиною в 28 байт команда обращения к ППЗУ будет выглядеть так:

1 009632
|
|
|
|
Длина записываемой программы в байтах
Адрес начального полубайта программы
Признак команды обращения к ППЗУ

После определения команды последняя записывается в графе 5 табл. 5.3, строка 4.

3. Набрать на клавиатуре пульта адрес обращения к ППЗУ (1009632).

4. Нажать клавишу $A \uparrow$, предписывающую запомнить адрес обращения к ППЗУ, отображаемый на экране индикатора. Как и в режиме стирания, выполнение команды $A \uparrow$ сопровождается отображением знаков минус во всех разрядах индикатора на фоне цифр команды.

5. Установить режим обмена программами между ОЗУП и ППЗУ, поставив переключатель Д — П в положение П (программа).

6. Установить ППЗУ в режим записи, для чего переключатель С — З — СЧ перевести в положение З (запись).

7. Нажать клавишу $\uparrow \downarrow$ (обмен между ОЗУП и ППЗУ), в результате чего произойдет запись программы из ОЗУП (с нулевого адреса) в поле памяти ППЗУ, границы которого определяются командой обращения к ППЗУ. В нашем случае поле памяти ППЗУ будет начинаться с 48-го байта (96-го полубайта) и заканчиваться 79-ым байтом (80-й байт — это уже начало следующей программы). При этом во всех разрядах индикатора будет отображаться знак минус.

8. Перевести ПМК в режим ввода программы, нажав клавиши FPRG. С помощью нажатия клавиш $\leftarrow \rightarrow$ убедиться в отсутствии на экране индикатора кодов команд записанной в ППЗУ программы, что свидетельствует о выполнении записи, так как после этого программа, находящаяся в ОЗУП, автоматически стирается.

9. Определить начало следующего (пятого) свободного поля и команду обращения к ППЗУ для его стирания:

$$Aб(5) = Aб(4) + Dб(4) = 048 + 32 = 080$$

$$Ac(5) = 2Aб(5) = 160.$$

Тогда команда для стирания следующего поля будет 1016098.

10. Произвести стирание поля ППЗУ, для чего набрать на пульте команду обращения к ППЗУ 1016098, перевести переключатель С — З — СЧ в положение С, нажать клавишу $A \uparrow$ и $\uparrow \downarrow$.

11. Поставить переключатель С — З — СЧ в положение СЧ (считывание). Это необходимо для защиты хранимых в ППЗУ программ от повреждений, которые могут быть в процессе работы на ПМК, если не отключить режимы стирания или записи.

На этом запись отложенной программы в ППЗУ заканчивается. Вместе с ее записью поле ППЗУ подготавливается к записи очередной программы.

П р е д у п р е ж д е н и е . 1. Стирание, запись и считывание отложенной программы из ППЗУ рекомендуется выполнять только при питании ПМК от сети через выпрямитель. Иначе емкость автономного источника будет быстро исчерпана из-за большого потребления энергии при обращении к ППЗУ.

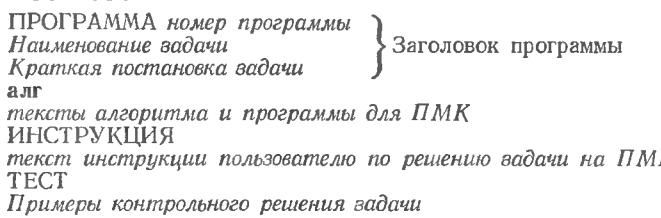
2. Запись программы в ОЗУП (при считывании ее из ППЗУ) и считывание программы из ОЗУП (при записи ее в ППЗУ) производится всегда с нулевого байта ОЗУП (нулевого адреса программы).

5.5. ОФОРМЛЕНИЕ ДОКУМЕНТАЦИИ НА ПРОГРАММУ

Развернутая форма документации. Как показывает практика программирования, разработанная и отложенная программа через некоторое время забывается даже автором и не может быть понята без затрат труда, сопоставимых с разработкой новой аналогичной программы. Поэтому отложенная программа сразу же после окончания отладки должна быть документирована, т. е. оформлена в виде, пригодном для ее дальнейшего использования. Для этого пользователю ПМК следует завести специальную тетрадь. Документация на каждую программу должна включать:

- 1) постановку задачи;
- 2) алгоритм и программу;
- 3) инструкцию;
- 4) тест (контрольное решение).

Совокупность перечисленных документов, строго говоря, является документацией на решаемую задачу, а не на программу для ПМК, которая сама по себе еще не обеспечивает решения задачи. Но следуя установившейся традиции и понимая программирование в широком смысле слова, будем называть эту документацию документацией на программу. Для ПМК все перечисленные документы целесообразно объединить в один, имеющий следующую структуру:



В этот документ включаются результаты всех этапов разработки программы, однако на этапе оформления документации они могут быть уточнены и дополнены по результатам отладки. Примерами оформления документации на программу могут служить программы 5.1 и 5.2.

Первый раздел сводного документа ПРОГРАММА устанавливает регистрационный ее номер, наименование задачи и краткую ее постановку. После отладки программы может возникнуть необходимость в уточнении постановки задачи в части дополнения и уточнения ограничений, сформулированных на первом этапе разработки программы, что иногда выявляется в процессе ее отладки. Номер программы устанавливается автором (пользователем ПМК). Концом первого раздела является начало второго — слово алг.

Во втором разделе помещается алгоритм решения задачи на всех трех уровнях его представления, включая и программу в виде последовательности команд ПМК. Концом второго раздела служит начало третьего — ИНСТРУКЦИЯ.

Раздел ИНСТРУКЦИЯ представляет собой часть общего алгоритма решения задачи, устанавливающую строгую последовательность действий пользователя в процессе решения задачи, начиная от ввода исходных данных и кончая получением исходных результатов. Инструкция включает в себя только предписания, адресованные пользователю. Никаких других предписаний, адресованных ПМК, или каких-либо других сведений инструкция не содержит. Она разрабатывается на основе постановки задачи, алгоритма ее решения и опыта решения задачи по составленной программе и составляется после окончания отладки программы. Для записи

предписаний инструкции используются команды алгоритмического языка первого уровня. Концом инструкции служит начало раздела ТЕСТ.

В разделе ТЕСТ приводится, по крайней мере, один пример контрольного решения в соответствии с приведенным алгоритмом, программой и инструкцией. В необходимых случаях в этом разделе может быть помещен не один, а несколько примеров решения задачи с различными вариантами исходных данных и получаемыми при этом результатами, например для нескольких ветвей решения задачи. Назначение раздела ТЕСТ — убедиться в работоспособности введенной в память ПМК программы, что всегда необходимо перед решением конкретной задачи по исходным данным пользователя. Эта цель может быть достигнута с помощью более простых операций по сравнению с операциями, выполняемыми пользователем при решении задачи. Учитывая, что в разделе ТЕСТ может быть несколько примеров, последовательность предписаний указанного раздела в большинстве случаев не совпадает с последовательностью предписания раздела ИНСТРУКЦИЯ. В разделе ТЕСТ должно быть также ориентировано указано ожидаемое время решения каждого из примеров (помещается в скобках после окончания записи примера: секунды — с, минуты — мин, часы — ч).

Рассмотренная форма записи документации на программу является ее развернутой формой. Такая форма удобна при разработке, отладке и сопровождении программы. На этапе ее сопровождения часто возникает необходимость в уточнении постановки задачи, модернизации алгоритма и программы с целью адаптации их к изменившимся условиям вычислений. Документация на разработанные и отлаженные программы, записанная в развернутой форме, записывается в тетрадь. Это — личный фонд программ пользователя ПМК (в него, как правило, входят алгоритмы и программы решения задач, им разработанные).

Компактная форма записи документации. Когда программа апробирована в процессе многократного решения и уже не нужно разбираться в ее тонкостях, развернутая форма документации может оказаться слишком громоздкой и неудобной. Для решения задач по отлаженным программам используется более удобная, компактная форма записи документации. Суть ее заключается в следующем:

1. В компактную форму записи документации включается только минимум сведений, необходимых для выполнения вычислений по готовой и отлаженной программе. Все другие сведения исключаются.

2. Раздел алг заменяется разделом КОМАНДЫ, в котором алгоритм решения задачи записывается только в одном (третьем) уровне — уровне программы, непосредственно выполняемой на ПМК. Команды программы записываются по возрастанию адресов столбцами (5—6 столбцов по ширине страницы). Каждая команда

записывается с указанием ее адреса, что облегчает ввод программы. Состояние стека после выполнения каждой команды не записывается.

Вместе с командами программы по ходу ее выполнения в соответствующих столбцах записываются команды алгоритма первого уровня, предписывающие пользователю выполнить необходимые действия. Эти команды записываются только в сокращенной форме: ВВ (ВВЕСТИ), ВЫ (ВЫДАТЬ), НЖ (НАЖАТЬ). Допускается использование команд ветвления, если передача управления в ту или иную ветвь производится пользователем вручную. Кроме того, при наличии тригонометрических функций используется команда установки их аргумента РАДИАНЫ — ГРАДЫ — ГРАДУСЫ. Запись команд ВВ и ВЫ совместно с командами программы позволяет определить аргументы и результаты программы.

3. В компактной форме записи документации исключается раздел ИНСТРУКЦИЯ, поскольку в разделе КОМАНДЫ уже записываются все предписания пользователю, содержащиеся в инструкции.

4. Заголовок программы и раздел ТЕСТ в компактной форме существенных изменений не претерпевают. Эти разделы по возможности записываются более кратко и компактно по всей ширине строки.

В качестве примера дадим запись программы 5.2 — расчета амплитуды и фазы результирующего колебания, но в компактной форме:

ПРОГ 5.2 К. Амплитуда $A = \sqrt{1 + n^2 + 2n \cos \psi}$, где $n = A_2/A_1$, $A_1 = 1$ и фаза $\phi = \operatorname{arctg}(n \sin \psi / (1 + n \cos \psi))$ — суперпозиции двух гармонических колебаний с амплитудами A_1 и A_2 и сдвигом фаз ϕ .

КОМАНДЫ
РАДИАНЫ 03. + 11. F V 17. 1 то НЖ 26. Ftg⁻¹
ВВ R0 = ψ 04. 2 12. C/P 18. + C/P 27. C/P
R1 = n 05. пX1 ВЫ A 19. C/P 20. пX1 ВЫ φ
НЖ B/O 06. × НЖ C/P 21. пX0 иначе
C/P 07. пX0 13. пX1 1+n cos ψ 22. F sin ВЫ φ = π/2
00. I 08. F cos 14. пX0 если 23. X все
01. пX1 09. X 15. F cos 1+n cos ψ 24. ↔
02. Fx² 10. + 16. X ≠ 25. ÷
ТЕСТ: ВВ R0 = ψ = 3π/2, R1 = n = 2, НЖ В/O С/P; ВЫ A = 2.2360679 (5 с)
НЖ С/P; ВЫ RX = I (3 с); НЖ С/P; ВЫ φ = -1.1071487 (5 с)

Компактная запись программы решения задачи может быть размещена на карточке. Пользователю рекомендуется завести картотеку или тетрадь для записи в компактной форме задач. Эта тетрадь-карточка будет представлять собой фонд программ, решаемых пользователем. В отличие от ранее упомянутой тетради, сюда будут записываться не только программы, разработанные пользователем, но и программы из печатных сборников или взятые у других пользователей.

Упражнение

5.5.1. Составить документацию на программу вычисления площади треугольника по формуле Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$ в компактной форме.

5.6. РЕШЕНИЕ ЗАДАЧИ ПО ДОКУМЕНТИРОВАННОЙ ПРОГРАММЕ

Для решения задачи по документированной программе как в развернутой, так и в компактной форме записи необходимо:

1. Сбросить счетчик адреса команд, нажав клавиши В/О (если пусковой адрес не равен 00, то набрать его с помощью команды БПн, где *nn* — пусковой адрес).

2. Ввести программу в память ПМК. Для этого:

1) Перевести ПМК из режима вычислений в режим ввода программы, нажав клавиши ПРГ.

2) Последовательно нажимать клавиши, записанные в программе в порядке возрастания их адресов. При вводе программы все команды, адреса которых кратны 10, а именно 00, 10, 20 и т. д., должны сверяться с адресами, отображаемыми на экране индикатора. При их несовпадении следует повторить ввод с той команды, вводимый и отображаемый на экране адрес которой совпадали.

3) Произвести испытание (тестирование) программы. Совпадение полученных результатов с указанными в teste свидетельствует о том, что программа введена в ПМК правильно и удостоверяет факт ее нормального функционирования. Если полученные результаты не совпадают с приведенными в teste, то следует повторить ввод программы, тщательно проверив ввод каждой команды, особенно команды Пхн и хПн, которые часто путаются.

5.6.1. СЧИТЫВАНИЕ ПРОГРАММЫ ИЗ ППЗУ В ОЗУП (ДЛЯ МК-52)

Если ранее программа была записана в ППЗУ МК-52, то можно не вводить ее в программную память путем набора на клавиатуре, а вместо этого произвести ее считывание из ППЗУ. Для этого после включения ПМК необходимо, поставив его в режим вычислений (нажав клавишу FABT), выполнить следующие операции:

1. Прочитать в таблице распределения памяти (см. табл. 5.3) команду обращения к программе, которую мы собираемся считывать. В случае программы А, φ (строка 4, графа 5 табл. 5.3) это будет команда 1009632.

2. Установить переключатель Д — П в положение П (программы), а С — З — СЧ — в положение СЧ (считывание).

3. Нажать клавишу А↑ (запоминание адреса обращения), а затем клавишу ↑↓ (обмен между ОЗУ и ППЗУ). При этом произойдет считывание программы из ППЗУ в ОЗУП. Считывание сопровождается появлением во всех разрядах индикатора знаков минус.

4. Перевести ПМК в режим ввода программы, нажав клавиши FПРГ. Просматривая коды команд на индикаторе с помощью нажатия клавиш ШГ и ШГ, убедиться в наличии программы в ОЗУП.

5.6.2. РЕШЕНИЕ ЗАДАЧИ ПО ВВЕДЕНОЙ В ОЗУП ПРОГРАММЕ

После ввода в ОЗУП программы можно приступить к решению задачи по исходным данным, для которых необходимо ее решить.

Произведем расчет амплитуды и фазы результирующего колебания по программе 5.2 для $\psi = 0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4, 3\pi/2, 7\pi/4, 2\pi$ и $n = 1, 2$. Результаты расчета сведем в табл. 5.4.

Может показаться, что предложенный способ разработки алгоритмов и программ для ПМК и оформления на них документации излишне громоздок. Действительно, в случае разработки линейных программ для ПМК можно было бы обойтись и двумя уровнями представления алгоритма: на алгоритмическом языке (лексиконе) (А-команды) и на входном языке ПМК (П-команды), исключив при этом команды БЕЙСИК-алгоритма. Однако, как будет показано ниже, для более сложных задач с ветвлениями и повторениями (реализуемыми не вручную пользователем, а выполняемыми автоматически на ПМК), многоуровневая разработка алгоритма и программы вполне себя оправдывает. Последовательная детализация алгоритма при многоуровневой его разработке уменьшает вероятность ошибок, а следовательно, экономит время на их поиск и устранение. В процессе сопровождения программы упрощается также введение в нее изменений и дополнений. Все это дает выигрыш во времени, намного превосходящий дополнительные затраты на разработку алгоритма и программы. Именно так в настоящее время разрабатываются алгоритмы и программы решения сложных задач на ЭВМ (метод структурного программирования). В предлагаемом способе разработки алгоритмов и программ для ПМК команды алгоритмов первого и второго уровней могут рассматриваться как обязательные рубрики алгоритмов третьего уровня — программы для ПМК, дисциплинирующие разработку последней и облегчающие ее дальнейшую эксплуатацию.

Таблица 5.4

Результаты расчета амплитуды A и фазы ϕ

ψ	$n = 1$		$n = 2$	
	A	ϕ	A	ϕ
1	2	3	4	5
0	2	0	3	0
$\pi/4$	1.847759	$3.9269909 - 01$	2.7979326	$5.22990281 - 01$
$\pi/2$	1.4142135	0.7853981	2.2360679	1.1071486
$3\pi/4$	7.6536669 - 01	1.1780972	1.4736258	-1.2858723
π	0	$\pi/2$	1	0
$5\pi/4$	7.6536677 - 01	-1.1780973	1.4736256	-1.285872
$3\pi/2$	1.4142135	-0.7853982	2.2360679	-1.1071487
$7\pi/4$	1.847759	-0.3926992	2.7979326	-0.5299029
2π	2	0	3	0

Упражнение

5.6.1. Выполнить решение по документированной программе вычисления площади треугольника по формуле Герона

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

где $p = (a+b+c)/2$, для следующих значений a, b, c :

a	b	c
0.5	1.00	1.2
1.00	2.00	2.75
5.00	2.00	2.75

Глава 6. ПРОГРАММИРОВАНИЕ ВЕТВЛЕНИЙ И ПОВТОРЕНИЙ

В предыдущей главе мы занимались автоматизацией вычислений на ПМК структур типа следования. Структуры типа ветвления и повторений выполнялись при этом вручную. Для дальнейшего сокращения затрат времени на выполнение вычислений и уменьшения вероятности ошибок необходимо как можно больше ручных операций, выполняемых пользователем вручную, передать для выполнений ПМК, в этом числе и такие операции как ветвления и повторения. При этом существенно усложняется программа решения задачи, зато значительно упростится самое решение и уменьшится вероятность появления ошибок в процессе решения из-за неправильных действий пользователя. Усложненная программа уже не будет линейной: она будет предписывать автоматическое выполнение ветвлений и повторений. Нелинейная программа будет включать в себя не только команды выполнения арифметических вычислений, но и команды управления вычислительным процессом, которые при ручном счете не используются. Отсюда значительное усложнение разработки нелинейных программ по сравнению с линейными, обусловленное необходимостью программирования ветвлений и повторений.

В настоящей главе будут рассмотрены особенности и примеры программирования ветвлений и повторений. Кроме того, на примере задачи расчета таблицы для построения гистограммы распределения случайной величины будет рассмотрено использование команд с косвенной адресацией. Указанные команды применяются в тех случаях, когда конкретные адреса регистров или команд программы, к которым необходимо обратиться в ходе решения задачи, еще неизвестны во время разработки программы и их требуется определить в процессе ее выполнения.

6.1. ПРОГРАММИРОВАНИЕ ВЕТВЛЕНИЙ

6.1.1. РАЗРАБОТКА АЛГОРИТМОВ ВЕТВЛЕНИЙ

Особенности алгоритмов ветвлений на языке БЕЙСИК-ПМК. Как уже отмечалось, в алгоритмическом языке первого уровня

представления ветвление описывается с помощью команды если — то — иначе. При описании ветвлений на втором уровне представления алгоритма, т. е. на языке БЕЙСИК-ПМК, должны учитываться особенности входного языка, структуры его команд, с помощью которых выполняются ветвления, а также особенности реализации последних в программе, иначе не будет обеспечено соответствие между БЕЙСИК-командами и командами программы, а следовательно, и формальное преобразование первых во вторые. Эти особенности можно свести к следующему:

1. В системе команд ПМК нет команды, непосредственно реализующей команду ветвления исходного алгоритма первого уровня его представления. Программы ветвления реализуются на ПМК не одной, а целой последовательностью команд, использующих команды условного и безусловного переходов. Эти команды в отличие от команд ветвления исходного алгоритма указывают адрес перехода, т. е. адрес той команды, к выполнению которой следует перейти, но не описывают все действия, которые надо выполнить в процессе выполнения ветвления. Кроме того, во всех командах условного перехода, предусмотренных в ПМК, указывается не адрес начала ветви то, а адрес начала ветви иначе. Поэтому говорят, что система команд ПМК предусматривает обратный синтаксис условного и безусловного переходов.

2. Командами ПМК предусмотрена проверка только простых условий. Сложные условия программируются с помощью последовательности команд, предусматривающих отдельную проверку всех простых условий, входящих в состав сложных.

3. Командами ПМК предусмотрена проверка лишь тех условий, правая часть которых равна нулю. Это требование нетрудно выполнить путем переноса с обратным знаком всех членов выражения, стоящего в правой части условия в левую его часть.

4. Командами ПМК предусмотрено, что в левой части условия должно быть помещено только имя переменной, значение которой является содержимым регистра X. В эквивалентной БЕЙСИК-команде это будет имя только одной переменной RX. Указанное требование также нетрудно выполнить путем помещения перед командой условного перехода серии команд, вычисляющих значение выражения, стоящего в левой части условия, правая часть которого равна нулю. В БЕЙСИК-алгоритме эта операция реализуется с помощью команды присваивания, помещаемой перед командой условного перехода.

5. Система команд ПМК обеспечивает возможность проверки не всех, а только следующих четырех условий: $x < 0$, $x \geq 0$, $x = 0$ и $x \neq 0$. Это означает, что и в БЕЙСИК-командах могут быть использованы только эти четыре условия. Если решаемая задача требует проверки условий $x \leq 0$ и/или $x > 0$, то их следует рассматривать как сложные, состоящие из двух простых условий. В частности, условие $x \leq 0$ может быть представлено как $x < 0$ или $x = 0$, а $x > 0$ — как $x \geq 0$ и $x \neq 0$. В исходном алгоритме

первого уровня представления для упрощения разработки программы рекомендуется использовать только условия, имеющие знаки $<$, \geq , $=$, \neq , а условия со знаками $>$ и \leq — представлять сложными.

Как уже отмечалось, процесс разработки алгоритма решения задачи, ориентированного на использование ПМК в программировании режиме, представляет собой составление исходного алгоритма с последующей детализацией его на втором и третьем уровнях представления. При этом команде исходного алгоритма первого уровня будет соответствовать, как правило, не одна, а серия команд второго уровня. Соответственно, команде второго уровня (БЕЙСИК-команде) будет соответствовать серия команд программы. Иными словами, каждая команда алгоритма старшего уровня представления интерпретируется алгоритмом, составленным из команд младшего уровня. Алгоритм преобразования команды старшего уровня в серию команд младшего уровня будем называть шаблоном команды. Следовательно, каждая команда алгоритмического языка (А-команда) будет иметь свой шаблон преобразования ее в серию БЕЙСИК-команд, каждая из которых также будет иметь свой шаблон преобразования в серию команд программы.

Весь процесс составления алгоритма второго и третьего уровней будем называть детализацией исходного алгоритма первого уровня. При наличии шаблонов на все типы команд алгоритмов первого и второго уровней детализация исходного алгоритма значительно упрощается и весь этот процесс становится почти полностью формализованным.

БЕЙСИК-команды, используемые при составлении алгоритмов ветвлений. В первом приближении будем полагать, что команда ветвления исходного алгоритма содержит только простое условие типа *выражение 1 знак выражение 2*. Оно легко приводится к условию типа *выражение знак 0*. Кроме того, условимся, что знаками условий могут быть только $<$, \geq , $=$, \neq . Условие вида *выражение знак 0* (где знак — это $<$, \geq , $=$, \neq) здесь и в дальнейшем будем называть каноническим.

Для того чтобы интерпретировать условие исходного алгоритма БЕЙСИК-командами, необходимо привести его к каноническому виду. Такое приведение осуществляется с помощью БЕЙСИК-команды присваивания: *номер RX = выражение 1 — выражение 2*. Результат выполнения такой команды всегда помещается в регистр X, даже если в левой части команды будет помещена одна из переменных $R0 \div RE$.

Для составления алгоритмов ветвлений в БЕЙСИК-ПМК используют команды условного и безусловного переходов. Команда условного перехода имеет структуру

номер IF RX знак 0 ELSE номер перехода.

где *номер перехода* — это номер команды (или строки), к выполнению которой предписывается перейти; IF — если, ELSE — иначе.

Команда условного перехода в зависимости от содержимого регистра X значения переменной RX передает управление либо следующей команде (ветвь *то*), либо БЕЙСИК-команде, номер которой указан после слова ELSE (ветвь *иначе*).

Кроме команды условного перехода для составления БЕЙСИК-алгоритмов ветвлений используется команда безусловного перехода, имеющая структуру

номер GOTO номер перехода.

где GOTO — иди к

Команда безусловного перехода предписывает приступить к выполнению команды, номер которой указан после слова GOTO, и отказаться при этом от выполнения команд, записанных непосредственно вслед за командой безусловного перехода. В отличие от команды условного перехода в рассматриваемой команде переход к выполнению команды с указанным номером производится во всех случаях, независимо от выполнения каких-либо условий.

Команды условного и безусловного переходов могут предписывать переход как вверх, к команде с меньшим номером, так и вниз, к команде с большим номером.

Шаблон команды ветвления

1. Шаблон команды ветвления в полной форме:

```
если A-выражение 1 знак A-выражение 2
      A RX = Б-выражение 1 — Б-выражение 2
      A + 1 IF RX знак 0 ELSE C + 1
    то A-серия 1
      B Б-серия 1
      C GOTO D
    иначе A-серия 2
      C + 1 Б-серия 2
    все
      D следующая Б-команда
```

Здесь и в дальнейшем при описаниях шаблонов будем пользоваться следующими обозначениями:

A-выражение, A-команда, A-серия — выражение, команда, серия команд исходного алгоритма, записанных на алгоритмическом языке (лексиконе);

Б-выражение, Б-команда, Б-серия — выражение, команда, серия команд, записанных на языке БЕЙСИК-ПМК.

Номера БЕЙСИК-команд обозначаются большими латинскими буквами *A, B, C, D...* Номер следующей команды обозначается *A + 1, B + 1, C + 1...* Если длина серии Б-команд не определена, то номер следующей команды обозначается другой латинской буквой:

```
... арг R0 = ψ, R1 = n
    рез RX = φ
    1 + n cos φ ≠ 0
если 1 RX = 1 + R1 * COS (R0)
      2 IF RX ≠ 0 ELSE 5
      то φ = n sin ψ/(1 + n cos ψ)
      3 RX = R1 * SIN (R0)/RX
      4 GOTO 6
    иначе φ = π/2
    5 RX = PI/2
все 6 следующая Б-команда
```

В приведенном примере заголовок команды ветвления если $1 + n \cos \phi \neq 0$ детализируется двумя БЕЙСИК-командами. Первая из них (команда присваивания) вычисляет значение выражения $1 + n \cos \phi \neq 0$, стоящего в левой части условия, с тем чтобы привести его к каноническому виду RX знак 0, а вторая является командой условного перехода. Она выполняется следующим образом. Если содержимое регистра X не равно нулю ($RX \neq 0$, что соответствует $1 + n \cos \phi \neq 0$), то выполняется следующая БЕЙСИК-команда: 3 RX = R1 * SIN (R0)/RX, соответствующая команде $\phi = n \sin \psi / (1 + n \cos \psi)$ исходного алгоритма. После выполнения этой команды выполнение ветви *то* заканчивается. С помощью БЕЙСИК-команды 4 GOTO 5 происходит выход из ветвления за служебное слово *все* и начинается выполнение следующей за ветвлением Б-команды 6. Если же условие $RX \neq 0$ не соблюдается ($RX = 0$ или $1 + n \cos \phi = 0$), то выполняется ветвь *иначе* (А-команда $\phi = \pi/2$, Б-команда 5 RX = PI/2). Затем происходит выход из ветвления и выполнение следующей за ветвлением БЕЙСИК-команды 6. Обратим внимание на то, что в отличие от ветви *то*, ветвь *иначе* не заканчивается БЕЙСИК-командой безусловного перехода GOTO. Выход из ветвления после выполнения ветви *иначе* происходит автоматически в порядке возрастания номеров команд, после окончания выполнения последней команды ветви *иначе*.

П р и м е ч а н и е. Для уменьшения вероятности ошибок при составлении алгоритмов ветвлений рекомендуется вначале не записывать в командах номера переходов, а вместо них ставить карандашом многоточие. И лишь тогда, когда все команды ветвления (кроме номеров переходов) будут составлены и записаны, вместо многоточий можно проставить в командах номера переходов, еще раз проверив при этом логику выполнения ветвления с помощью БЕЙСИК-команд.

Как видно из рассматриваемого примера, при описании ветвлений с помощью БЕЙСИК-команд широко используются текущая переменная RX, значением которой является содержимое регистра X, отображаемое на экране индикатора. При этом следует учитывать, что эта переменная, вычисленная в предыдущей команде присваивания 1 RX = 1 + R1 * COS (R0), сохраняет свое значение и в последующей команде присваивания 3 RX = ATG (R1 * SIN (R0)/RX). Здесь значение RX в знаменателе аргумента тригонометрической функции представляет собой результат вычис-

ления предыдущей команды присваивания 1. После выполнения команды присваивания 3 RX приобретает новое значение (старое значение в данном случае теряется). Если между командами присваивания помещаются команды условного и/или безусловного переходов, то выполнение последних не влияет на изменение значения RX. По этой причине команда условного перехода 2 не влияет на значение RX и оно может быть использовано для выполнения команды присваивания 3.

2. Шаблон сокращенной формы команды ветвления:

```

если A-выражение 1 знак A-выражение 2
    А   RX = Б-выражение 1 — Б-выражение 2
    А+1 IF RXзнак 0 ELSE С
    то A-серия
    все
если 1 + n cos ψ ≠ 0
    1   RX = 1 + R1 * COS (R0)
    2   IF RX ≠ 0 ELSE 4
    то ϕ = n sin ψ / (1 + n cos ψ)
    3   RX = R1 * SIN (R0) / RX
    4   следующая Б-команда
    все

```

В отличие от полной формы шаблон сокращенной формы команды ветвления не содержит команды безусловного перехода в конце ветви **то**, так как в этом нет необходимости. В самом деле, после выполнения последней команды ветви **то** команды 3 происходит автоматический переход к выполнению команды 4, находящейся уже вне ветвления.

Шаблоны ветвлений со сложными условиями

Выше было отмечено, что системой команд ПМК не предусмотрена проверка выполнения условий $x < 0$ и $x > 0$. Для обеспечения возможности проверки таких условий необходимо представить их в исходном алгоритме как сложные, состоящие из двух простых условий. Для упрощения представления шаблонов будем полагать, что уже в исходном алгоритме они приведены к каноническому виду, т. е. простые условия, входящие в состав сложных, имеют вид: **имя знак 0**.

1. Шаблон команды ветвления со сложным условием типа и:

```

если условие 1 и условие
    А   IF RXзнак 0 ELSE D
    А+1 IF RXзнак 0 ELSE D
    то A-серия 1
    все
    иначе A-серия 2
    Е   следующая Б-команда

```

Пример:
 Разработать алгоритм вычисления функции $y = \lg x$. При невозможности вычислений выдать $y = 1.111111 - 01$.

Решение. Вычисление y возможно при $x > 0$. Это условие можно рассматривать как требование одновременного выполнения условий $x \geq 0$ и $x \neq 0$:

алг логарифм * $y = \lg x *$
 арг RX = x
 арг $RX = y$
 вач
 ВВ RX = x
 если $x \geq 0$ и $x \neq 0$

- 1 IF RX ≥ 0 ELSE 5
- 2 IF RX $\neq 0$ ELSE 5
- то $y = \lg x$
- 3 RX = LG (RX)
- 4 GOTO 6
- иначе $y = 1.111111 - 01$
- 5 RX = 1/9
- все
- Вы y
- 6 STOP

 кон

Здесь команда условного перехода 1 проверяет условие $RX \geq 0$. Если оно соблюдается, то происходит переход к выполнению второй БЕЙСИК-команды, проверяющей условие $RX \neq 0$. Если и это условие соблюдается, то произойдет переход к выполнению третьей БЕЙСИК-команды, вычисляющей $\lg x$. Таким образом, вычисление $y = \lg x$ производится только при одновременном выполнении обоих условий: $RX \geq 0$ и $RX \neq 0$, что эквивалентно одному условию: $RX > 0$. В данном алгоритме ветвь **то** включает в себя БЕЙСИК-команды 3 и 4, а ветвь **иначе** — БЕЙСИК-команду 5. Командой, непосредственно следующей за ветвлением, является команда ВЫДАТЬ исходного алгоритма, а эквивалентной ей — БЕЙСИК-команда 6, предусматривающая останов вычислений для выдачи результатов. Особенностью алгоритма является не простое присваивание константы $y = 1.111111 - 01$, а ее вычисление с помощью БЕЙСИК-команды 7 RX = 1/9. Это дает возможность интерпретировать эту команду лишь двумя командами ПМК: $a + 0.9$ и $a + 1. F1/x$, в то время как для реализации БЕЙСИК-команды 7 RX = 1.111111 - 01 потребовалось бы 11 команд программы.

2. Шаблон команды ветвления со сложными условиями типа или:

```

если условие 1 или условие 2
    А   IF RXзнак 0 ELSE С
    А+1 GOTO А+3
    А+2 IF RXзнак 0 ELSE С
    то A-серия 1
    все
    иначе A-серия 2
    А+3 Б-серия 1
    А+4 GOTO D
    Б   Б-серия 2
    С   Б-серия 2
    Д   следующая Б-команда

```

Команды ПМК, используемые для программирования ветвлений

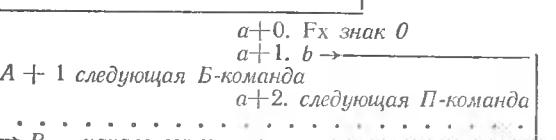
Команда	Выполняемые действия
БП <i>nn</i>	Безусловный переход к выполнению команды с адресом <i>nn</i>
$Fx \geq 0$ <i>nn</i>	Переход к выполнению следующей команды программы при выполнении условия ($X \geq 0$). При его невыполнении переход к выполнению команды с адресом <i>nn</i>
$Fx = 0$ <i>nn</i>	При ($X = 0$) переход к выполнению следующей за <i>nn</i> команде программы, при ($X \neq 0$) переход к выполнению команды с адресом <i>nn</i>
$Fx \neq 0$ <i>nn</i>	При ($X \neq 0$) переход к выполнению следующей за <i>nn</i> команде программы, при ($X = 0$) переход к выполнению команды с адресом <i>nn</i>

Как уже указывалось, системой команд ПМК предусмотрена проверка условий ($X \geq 0$), ($X < 0$), ($X = 0$), ($X \neq 0$), где (X) — содержимое регистра X в момент проверки. К этим командам по своей структуре примыкает команда безусловного перехода БП*nn*, осуществляющая переход к указанному в ней адресу *nn* независимо от соблюдения каких-либо условий. Команды ПМК, используемые для программирования ветвлений, приведены в табл. 6.1.

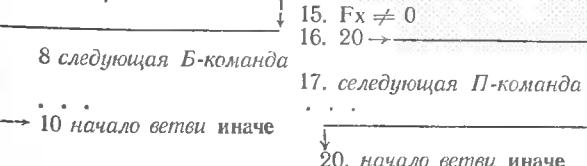
6.1.3. ШАБЛОНЫ БЕЙСИК-КОМАНД

Шаблон команды условного перехода

A + 0 IF RX знак 0 ELSE B →



7 IF RX ≠ 0 ELSE 10 →



Разработать алгоритм вычислений функции $y = \sqrt{-x}$. При невозможности вычислений выдать $y = 1.111111 - 01$.

Решение. Здесь вычисления возможны при $x \leq 0$. Это условие можно представить как сложное: $x < 0$ или $x = 0$. Тогда, рассматривая сложное условие как комбинацию двух простых, можно записать алгоритм вычисления функции $y = \sqrt{-x}$ в следующем виде:

```

алг корень
арг RX = x
рез RX = y
нач
  ВВ x
  если x < 0 или x = 0
    1 IF RX < 0 ELSE 3
    2 GOTO 4
    3 IF RX = 0 ELSE 6
    то y = √-x
      4 RX = SQR (-RX)
      5 GOTO 7
    иначе y = 1.111111 E - 1
      6 RX = 1/9
    все
    ВЫ y
    7 STOP
  
```

В отличие от предыдущего алгоритма здесь простые условия соединяются служебным словом или (а не и). В этом случае детализация заголовка исходного алгоритма будет осуществляться тремя (а не двумя) БЕЙСИК-командами. Первая из них осуществляет проверку первого условия и в случае его соблюдения с помощью БЕЙСИК-команды 2 GOTO 4 предписывает выполнить ветвь то. В случае несоблюдения первого условия управление передается БЕЙСИК-команде 3 IF RX = 0 ELSE 6, которая проверяет соблюдение второго условия. Если оно соблюдается, то выполняется ветвь то, если нет (а это имеет место, когда оба условия не соблюдаются), то выполняется ветвь иначе. В остальном порядок выполнения данного алгоритма аналогичен изложенному в примере 2.

6.1.2. КОМАНДЫ ПМК, ВЫПОЛНЯЮЩИЕ УСЛОВНЫЕ И БЕЗУСЛОВНЫЕ ПЕРЕХОДЫ

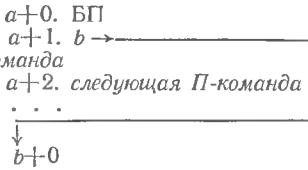
Для программирования ветвлений в ПМК предусмотрены две х-шаговые команды условного перехода типа

<i>F_p</i> условие (1-й шаг)	<i>nn</i> адрес (2-й шаг)
--	---------------------------------

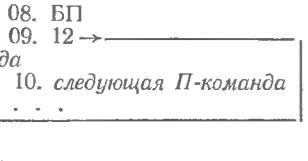
Если условие *p* выполняется, то осуществляется переход к выполнению следующей команды (3-й шаг). Если оно не выполняется, то осуществляется переход к выполнению команды с адресом, указанным на втором ее шаге, непосредственно вслед за условием.

Шаблон команды безусловного перехода

A GOTO B →



5 GOTO 8 →



6 следующая Б-команда

10. следующая П-команда

Как видим, команды условного и безусловного переходов в БЕЙСИК-алгоритме и программе ПМК по своей структуре мало отличаются друг от друга. Отличие состоит в том, что БЕЙСИК-команда записывается в одну строку, а соответствующая ей команда программы — в две строки, так как все команды условного и безусловного переходов в ПМК являются двухшаговыми. Команды условного и безусловного переходов позволяют осуществлять переход как вниз, когда $B > A$ и $b + 0 > a + 0$, так и вверх, когда $B < A$ и $b + 0 < a + 0$.

Пример программы с ветвлением

Рассмотрим программу расчета фазы результирующего колебания как пример программы с ветвлением. Используя приведенные выше шаблоны команд условного и безусловного переходов, составим программу вычисления фазы суммы двух гармонических колебаний.

ПРОГРАММА. 6.1. Расчет фазы суммы двух гармонических колебаний:

$$\varphi = \arctg((n \sin \psi) / (1 - n \cos \psi)),$$

где ψ — сдвиг фаз между составляющими гармоническими колебаниями; n — соотношение их амплитуд.

алг фаза

апр R0 = ψ , R1 = n
рез RX = φ

нач

BB R0 = ψ

если $1 + n \cos \psi \neq 0$

1 RX = $1 + R1 * \cos(R0)$

00. 1 (1)

01. пX1 (R1, 1)

02. пX0 (R0, R1, 1)

03. F cos (COS (R0), R1, 1)

04. X (R1 * COS (R0), 1)

05. + (1 + R1 * COS (R0))

2 IF RX ≠ 0 ELSE 5

06. Fx ≠ 0

07. 16

то $\varphi = \arctg(n \sin \psi / (1 + n \cos \psi))$

3 RX = ATG (R1 * SIN (R0) / RX)

08. пX1 (R1, 05RX)

09. пX0 (R0, R1, 05RX)

10. F sin (SIN (R0), R1, 05RX)

11. X (R1 * SIN (R0), 05RX)

12. ↔ (05RX, 11RX)

13. ÷ (11RX/05RX)

14. F tg⁻¹ (ATG (11RX/05RX))

4 GOTO 6

15. БП

16. 20

иначе $\varphi = \pi/2$

5 RX = P / 2

все

ВЫ φ

6 STOP

20. С/П

кон

ИНСТРУКЦИЯ BB R0 = ψ , R1 = n ; НЖ В/О С/П; ВЫ φ

ТЕСТ

1 BB R0 = $\psi = \pi/2$, R1 = $n = 1$; НЖ В/О С/П; ВЫ $\varphi = 0.7853981 - 01$

2 BB R0 = $\psi = \pi$, R1 = $n = 1$; НЖ В/О С/П; ВЫ $\varphi = \pi/2 = 1.5707963$

6.1.4. СОСТАВЛЕНИЕ АЛГОРИТМОВ ВЫБОРА НА ЯЗЫКЕ БЕЙСИК-ПМК

Шаблон команды выбора

В отличие от команды ветвления команда выбора алгоритмического языка предписывает разделение вычислительного процесса более чем на две ветви. Общий вид команды выбора приведен в п. 4.1.1. Для удобства записи шаблона команды выбора в первом приближении предположим, что все условия в ней являются простыми, а знаки в них ограничены знаками $>$, $<$, $=$, \neq . Тогда шаблон команды выбора будет иметь вид

выбор

при условие 1: A-серия 1

A+0 IF RX знак 0 ELSE C+0

A+1 Б-серия 1

B GOTO H

при условие 2: A-серия 2

C+0 IF RX знак 0 ELSE D+1

C+1 Б-серия 2

D GOTO H

при условие n A-серия n

E+0 IF RX знак 0 ELSE G

E+1 Б-серия n

F GOTO H

[иначе A-серия]

G Б-серия]

H следующая Б-команда

все

5*

Пример составления алгоритма выбора на языке БЕЙСИК-ПМК

Рассмотрим на примере, как команда выбора детализируется командами БЕЙСИК-ПМК. Пусть необходимо вычислить значение функции

$$y = \begin{cases} \sqrt{x} & \text{при } x \geq 0.5, \\ \cos x & \text{при } 0 < x < 0.5, \\ \sin x & \text{при } x \leq 0. \end{cases}$$

Вычисление указанной функции может быть описано с помощью команды выбора:

выбор

```
    при  $x \geq 0.5$ :  $y = \sqrt{x}$ 
    при  $x > 0$  и  $x < 0.5$ :  $y = \cos x$ 
    при  $x \leq 0$ :  $y = \sin x$ 
```

все

Прежде всего необходимо проанализировать команду выбора с целью упрощения дальнейших вычислений и обеспечения возможности их автоматизации на ПМК. Если первое условие $x \geq 0.5$ соблюдается, то вычисляется $y = \sqrt{x}$, после чего происходит выход из команды выбора. Если условие $x \geq 0.5$ не соблюдается (т. е. $x < 0.5$), то проверяется выполнение второго условия: $x > 0$ и $x < 0.5$. Так как условие $x < 0.5$ уже проверено на первом шаге выбора, то на втором шаге достаточно проверить только условие $x > 0$. Если последнее соблюдается, то вычисляется значение $y = \cos x$, а затем происходит выход из команды выбора. Если же условие $x > 0$ не соблюдается, т. е. если окажется, что $x < 0$, то происходит переход к третьему шагу выбора, на котором должно проверяться условие $x \leq 0$. Очевидно, что такая проверка становится излишней, так как при переходе к третьему шагу выбора условие $x \leq 0$ уже заведомо выполняется.

Проверка условия $x \leq 0$ командами ПМК не предусмотрена, и это существенно упрощает задачу. С учетом сделанных замечаний команда выбора может быть сведена к следующему виду:

выбор

```
    при  $x \geq 0.5$ :  $y = x$ 
    при  $x \geq 0$  и  $x \neq 0$ :  $y = \cos x$ 
    иначе  $y = \sin x$ 
```

все

Указанная команда исходного алгоритма может быть детализирована с помощью БЕЙСИК-команд условного и безусловного переходов согласно приведенному шаблону:

```
...     arg R0 = x
      рез RX = y
...
```

выбор

```
    при  $x \geq 0.5$ :  $y = \sqrt{x}$ 
          1 RX = R0 - 0.5
          2 IF RX  $\geq 0$  ELSE 5
```

```
3 RX = SQR (R0)
4 GOTO 11
при  $x \geq 0$  и  $x \neq 0$ :  $y = \cos x$ 
5 RX = R0
6 IF RX  $\geq 0$  ELSE 10
7 IF RX  $\neq 0$  ELSE 10
8 RX = COS (R0)
9 GOTO 11
иначе  $y = \sin x$ 
10 RX = SIN (R0)
```

все

11 следующая Б-команда

На первом шаге команды выбора вычисляется значение левой части условия с помощью БЕЙСИК-команды 1, а затем проверяется само условие — Б-команда 2. При его выполнении вычисляется значение $y = \sqrt{x}$ (Б-команда 3), а затем с помощью БЕЙСИК-команды 4 осуществляется переход к БЕЙСИК-команде 11, непосредственно следующей за командой выбора. Если условие не соблюдается, то осуществляется переход ко второму шагу выбора, к выполнению БЕЙСИК-команды 5, вычисляющей левую часть первого условия второго шага. БЕЙСИК-команды 6 и 7 интерпретируют сложное условие $x \geq 0$ и $x \neq 0$ простыми условиями. При их соблюдении вычисляется значение $y = \cos x$ (Б-команда 8), затем происходит выход из команды выбора с помощью БЕЙСИК-команды 9 (к Б-команде 11). Если условие не соблюдается, то выполняется БЕЙСИК-команда 10, предписывающая вычисление $y = \sin x$, после чего происходит выход из команды выбора.

Упражнения

6.1.1. Разработать программу вычисления на ПМК функции

$$Y = \begin{cases} \sqrt{x} & \text{при } x \geq 0.5, \\ \cos x & \text{при } 0 < x < 0.5, \\ \sin x & \text{при } x \leq 0. \end{cases}$$

6.1.2. Разработать программу вычисления на ПМК функции $y = \lg x$. При невозможности вычислений выдать признак $y = 1.1111111 - 01$.

6.1.3. Разработать программу вычисления на ПМК функции $y = \sqrt{x}$. При невозможности вычислений выдать признак $y = 1.1111111 - 01$.

6.2. ПРОГРАММИРОВАНИЕ ПОВТОРЕНИЙ

6.2.1. РАЗРАБОТКА АЛГОРИТМОВ ПОВТОРЕНИЙ

Особенности алгоритмов повторений на языке БЕЙСИК-ПМК

Алгоритмический язык, который мы выбрали для описания исходного алгоритма решения задачи (п. 4.1.1), включает в себя три типа команд повторения:

пока;
до;
с параметром.

Системой команд ПМК не предусмотрено непосредственное вычисление повторений. Они выполняются на ПМК сериями команд, включающими в себя команды условного и безусловного переходов. Соответственно и в языке БЕЙСИК-ПМК повторения также описываются с использованием БЕЙСИК-команд условного и безусловного переходов. Как и в случае ветвлений, предполагается, что условия, входящие в повторение исходного алгоритма, будут простыми (вида *выражение знак выражение*) и легко приводимыми к каноническому виду (RX знак 0), где допустимыми знаками являются $<$, $>$, $=$, \neq . Команды повторения исходного алгоритма всех перечисленных типов преобразуются в серии БЕЙСИК-команд с помощью шаблонов, которые будут рассмотрены ниже.

Помимо рассмотренных в п. 6.1.2 команд условного перехода системой команд ПМК предусмотрена также команда условного перехода по счетчику. Для ее применения необходимо в регистр, используемый в качестве счетчика, занести количество требуемых обращений к команде. При каждом обращении из содержимого счетчика вычитается единица. Таким образом, перед каждым обращением к команде содержимое счетчика показывает, сколько раз еще потребуется к ней обращаться. Когда содержимое счетчика станет равным единице, то выполняется команда, непосредственно следующая за командой условного перехода по счетчику. Пока содержимое счетчика не стало равным единице, осуществляется переход к выполнению команд, адрес которых указан в команде условного перехода по счетчику.

Для того чтобы использовать возможности ПМК по организации повторений с использованием счетчика, введем в состав команд алгоритмического языка первого уровня представления команду повторения по счетчику:

```

нц
| серия
кц
до X = 1 от Xкон шаг —1

```

Здесь $X_{\text{кон}}$ — заданное число повторений. Команда повторения по счетчику будет обеспечивать повторение серии команд, записанных между нц и кц $X_{\text{кон}}$ раз. Счет числа повторений здесь идет в обратном порядке: от конечного значения параметра $X = X_{\text{кон}}$ к начальному $X = 1$. После выполнения серии команд и проверки условия $X = 1$ из текущего значения X будет вычитаться единица, а затем снова повторяться выполнение серии команд. Как только значение X станет равным единице, произойдет выход из команды повторения, т. е. переход к выполнению следующей за повторением команды. Команду повторения по счетчику мы будем широко использовать в дальнейшем при описании исходных алгоритмов решения задач.

БЕЙСИК-команда условного перехода по счетчику

Для описания повторений по счетчику используется БЕЙСИК-команда условного перехода по счетчику:

номер UNTIL Rn = 1 [FROM Rn STEP —1] GOTO...,

где UNTIL — до; FROM — от; STEP — шаг; GOTO — иди к; n — номер регистра, в который помещается значение счетчика ($n = 0, 1, 2, 3$, т. е. в качестве счетчиков могут использоваться только регистры R0—R3).

Примечание. Служебные слова FROM Rn STEP —1 можно опустить, так как счет начинается всегда от значения Rn и уменьшается с шагом —1.

Команда выполняется следующим образом:

- 1) проверяется значение переменной Rn;
- 2) если оно не равно единице, то из него вычитается единица;
- 3) осуществляется переход к выполнению команды, номер которой указан после служебного слова GOTO;
- 4) как только переменная станет равной единице, осуществляется переход к выполнению следующей БЕЙСИК-команды.

Шаблон команды повторения типа пока

пока выражение 1 знак выражение 2
 A+0 RX = выражение 1 — выражение 2
 A+1 IF RX знак 0 ELSE D

нц А-серия
 | В Б-серия
 кц C GOTO A+0

Д следующая Б-команда

Пример. Вычислить сумму членов ряда
 $S = 1 + 1/x + 1/x^2 + 1/x^3 + \dots$
 при $x > 1$ с точностью до 0.001.

Обозначим:

$v[i]$ — очередной член ряда;
 $S[i]$ — текущая сумма членов ряда $v[0] + v[1] + v[2] + \dots + v[i]$.

Решение.

$$\begin{aligned} S[0] &= 0; \\ S[1] &= S[0] + v[0]; \\ S[2] &= S[1] + v[1]; \end{aligned}$$

$$S[i] = S[i-1] + v[i-1].$$

Параметром повторения, т. е. величиной, сохраняющей свое постоянное значение в пределах одного повторения и меняющейся при переходе к другому, будет здесь значение очередного члена ряда v , а функцией, с помощью которой вычисляется очередной член ряда из предыдущего, $v = v/x$. Тогда, полагая $v[0] = 1$, получим:

$$\begin{aligned} v[0] &= 1; \\ v[1] &= v[0]/x = 1/x; \\ v[2] &= v[1]/x = 1/x^2; \\ v[i] &= v[i-1]/x = 1/x^i. \end{aligned}$$

Отсюда следует, что вычисления должны быть прекращены и осуществлен выход из команды повторения, когда станет выполняться условие $v < 0.001$.

С учетом изложенного алгоритм вычисления суммы членов ряда примет следующий вид:

```
алг сумма ряда * S = 1 + 1/x + 1/x2 ... при x > 1 с точностью 0.001 *
    арг R0 = x
    рез RX = S
нач R1 = v, R2 = S
    BB RX = x; НЖК В/О С/П;
    v = 1
        1 R1 = 1
    S = 0
    пока v ≥ 0.001
        2 R2 = 0
        3 RX = R1 - 0.001
        4 IF RX ≥ 0 ELSE 8
        иц
            S = S + v
            v = v*x
        иц
            ВЫ S, v
        кон
        5 R2 = R2 + R1
        6 R1 = R1/R0
        7 GOTO 3
        8 RX = R2
        9 STOP
```

Возврат к повторной проверке условия и в случае его соблюдения к выполнению следующего цикла осуществляется с помощью команды безусловного перехода 7 GOTO 3, а выход из повторения — с помощью БЕЙСИК-команды 4 (по ветви ELSE — иначе) к БЕЙСИК-команде 8. Назначение последней — подготовка результата решения (суммы членов ряда S) к выдаче на индикатор. При этом значение S помещается в регистр X, а значение v перемещается из регистра X в регистр Y.

Шаблон повторения типа до

```
иц
    A-серия
    иц
        A Б-серия
    до выражение 1 знак выражение 2
        B RX = Б-выражение 1 — Б-выражение 2
        B+1 IF RX знак 0 ELSE A
        C следующая Б-команда
```

Пример. Вычислить сумму членов того же ряда, что и в предыдущем примере, но только в алгоритме использовать повторение не типа пока, а типа до.

Решение.

```
алг сумма ряда * Вариант 2. S = 1 + 1/x + 1/x2 + ...
    при x > 1 с точностью 0.001 *
    арг R0 = x
    рез RX = S
нач R1 = v, R2 = S
    BB R0 = x; НЖК В/О С/П
    v = 1
        1 R1 = 1
    S = 0
        2 R2 = 0
```

```
иц
    S = S + v
    3 R2 = R2 + R1
    v = v/x
    4 R1 = R1/R0
    иц
        до v ≤ 0.001
            5 RX = R1 - 1E - 3
            6 IF RX ≤ 0 ELSE 3
    ВЫ S, v
    7 RX = R2 (R2, R1)
    8 STOP
кон
```

Сравнивая шаблоны повторений типов пока и до, а также примеры 1 и 2 этого параграфа, можно заметить, что использование повторения типа до уменьшает длину алгоритма на одну БЕЙСИК-команду. В шаблоне повторения типа до отсутствует БЕЙСИК-команда безусловного перехода GOTO 3, обеспечивающая в шаблоне пока возврат к началу повторения для выполнения очередного цикла. Вместо нее в шаблоне повторения типа до эту операцию выполняет БЕЙСИК-команда условного перехода (в шаблоне B + 1, в примере 2—6), стоящая в конце команды повторения.

Таким образом, в тех случаях, когда для одной и той же задачи возможно использование повторений типа пока и до, использование последнего типа предпочтительно, так как при этом БЕЙСИК-алгоритм сокращается на одну команду, а программа для ПМК — на два шага (команда безусловного перехода в ПМК — двухшаговая БПпп).

Шаблон повторения с параметром

1. Шаблон повторения с определенным числом значений параметра:
... Ri = x, Rf = X_{нач}, Rg = X_{кон}, Rh = X_{шаг}
для X от X_{нач} до X_{кон} шаг X_{шаг}
A+0 Ri = Rf - Rh
A+1 Ri = Ri + Rh
A+2 RX = Rg - Ri
A+3 IF RX ≥ 0 ELSE C+1

```
иц
    A-серия
    иц
        A Б-серия
        C GOTO A+1
```

C+1 следующая Б-команда

Заголовок команды повторения исходного алгоритма детализируется четырьмя БЕЙСИК-командами. Первая из них предназначена для определения начального значения X: X = X_{нач} — X_{шаг}. Поскольку в каждом повторении, включая и первое, с помощью БЕЙСИК-команды A + 1 Ri + Rh к значению X будет прибавляться X_{шаг}, то X_{нач} будет соответствовать X = X_{нач} — X_{шаг} + X_{шаг}. Для последующих повторений в процессе выполнения рассматриваемой команды параметр X будет принимать следующие значения: X = X_{нач} + 2X_{шаг}, X_{нач} + 3X_{шаг} и т. д. Третья

БЕЙСИК-команда интерпретирует условие выхода из повторения. Если $X_{\text{кон}} - X \geq 0$, то происходит выполнение очередного повторения от иц до кц . Если же нет (ветвь *иначе*), то происходит выход из повторения — выполнение следующей БЕЙСИК-команды $C + 1$.

При мер. Вычислить сумму натурального ряда чисел от 1 до n .
алг СМНР * сумма чисел натурального ряда от 1 до n *

```

арг R1 = n
рез RX = S
нач R2 = i, R3 = S
BB R1 = n; НЖ В/О С/П
S = 0
    1 R3 = 0
для i от 1 до n
    2 R2 = 0
    3 R2 = R2 + 1
    4 RX = R1 - R2
    5 IF RX >= 0 ELSE 8
    |
    | нц S = S + i
    |
    | кц ВЫ S
    |
    | кон
    6 R3 = R3 + R2
    7 GOTO 3
    |
    8 RX = R3
    9 STOP

```

БЕЙСИК-команда 2 $R2 = 0$, интерпретирующая заголовок повторения, образуется из команды шаблона $A + 0 Ri = Rf - Rh$. Поскольку $Rf = 1$, $Rh = 1$, а $Ri = R2$, то $R2 = 0$.

2. Шаблон повторения с неопределенным числом значений параметра:

```

... Ri = X, Rf = X_нач Rh = X_шаг, таб RX = X
для X от X_нач до длин (X) шаг X_шаг
    [A+0 Ri = Rf - Rh
    A+1 Ri = Ri + Rh]
    A+2 STOP

```

```

нц
    A-серия
    |
    | В-серия
    | C GOTO A+1
    |
    кц

```

При мечание. БЕЙСИК-команды $A+0$ и $A+1$ могут выполняться пользователем вручную. В этом случае они могут быть опущены.

В шаблоне с неопределенным числом значений параметра заранее неизвестно, когда должно быть закончено повторение и осуществлен выход из него. Это решается в процессе вычислений самим пользователем в конце каждого повторения, для чего требуется БЕЙСИК-команда останова B STOP. Этот же останов используется и для ввода очередного значения X . Поскольку определение условия выхода из повторения возложено на пользователя ПМК, то нет необходимости в БЕЙСИК-команде условного перехода. Возврат к началу очередного повторения обеспечивает

БЕЙСИК-команда безусловного перехода C GOTO A + 1 на останов вычислений. Во время указанного останова пользователь примет решение, продолжать ли очередное повторение или выйти из него на выполнение следующей за повторением команды.

Пример. В результате опыта некоторая случайная величина x приняла значения $x[1] \dots x[i] \dots x[n]$. Определить среднее значение x этой величины.

Решение. Значения текущей величины $x[i]$ будем вводить вручную, и после ввода каждого i -го значения автоматически вычислять среднее значение всех ранее введенных величин, рассчитывая его по формуле

$$x = (1/i) \sum_{i=1}^l x[i] = S[i]/i.$$

В процессе вычислений для этого накапливается сумма всех введенных значений и подсчитывается их количество. Подсчет текущей суммы S и количества слагаемых i производится следующим образом:

$$\begin{array}{ll} S[0] = 0; & i[0] = 0; \\ S[1] = S[0] + x[0]; & i[1] = i[0] + 1; \\ S[2] = S[1] + x[1]; & i[2] = i[1] + 1; \\ \vdots & \vdots \\ S[i] = S[i-1] + x[i-1]; & i[k] = i[k-1] + 1. \end{array}$$

Подсчет производится с помощью команд исходного алгоритма $S = S + i$, $i = i + 1$. Конечное значение i при этом не будет определено, поэтому задача вычисления S и i решается с помощью шаблона повторения с неопределенным числом значений параметра. Алгоритм такого повторения будет иметь вид

```

алг среднес * x[i] = (1/i) \sum_{i=1}^l x[i] *
арг таб RX = x
рез RX = x
нач RO = i, R1 = S
S = 0
    1 R1 = 0
для i от 1 до длин (x)
    2 R0 = 0
    3 R0 = R0 + 1
|
    | нц
    |     БЫ S[i-1]
    |     |
    |     BB x[i]; НЖ В/О С/П
    |     S = x[i] + S
    |     x = S/i
    |
    |     кц
    |     4 STOP
    |
    |     5 R1 = RX + R1
    |     6 RX = R1/R0
    |     7 GOTO 3
|
кон

```

Начальное значение параметра i здесь выбрано равным нулю для того, чтобы удобнее было интерпретировать заголовок повторения, предписывающий увеличение параметра i на единицу с каждым вновь выполняемым циклом, включая и первый. Если бы мы задали в качестве начального значения $i = 1$, то БЕЙСИК-команду 3 пришлось бы поместить после Б-команды 6. Это не повлияло бы на результат решения задачи, но тогда внутри цикла имело бы место несовпадение между значениями i для исходного алгоритма и командами БЕЙСИК-алгоритма.

Шаблон повторения по счетчику

$A + 0 RX = Rn$

... $Rn = N$

иц
кц

А-серия

В Б-серия

до $N = 1$ от N шаг -1

$C \text{ UNTIL } Rn = 1 [\text{FROM } Rn \text{ STEP } -1] \text{ GOTO } B$

Здесь в регистр Rn (где $n = 0, 1, 2, 3 \dots$) перед началом выполнения команды повторения вводится значение количества повторений N . Этот ввод может быть осуществлен с пульта ПМК вручную и с помощью команды присваивания при автоматическом выполнении алгоритма.

После выполнения серии команд каждого цикла из текущего значения N вычитается единица, после чего осуществляется переход к началу следующего цикла. Как только текущее значение N станет равным единице, т. е. $Rn = 1$, осуществляется переход к следующей за повторением БЕЙСИК-команде.

Пример. Составление БЕЙСИК-алгоритма факториала $N!$.

Решение. Для вычисления факториала $N!$ необходимо перемножить N чисел натурального ряда от 1 до N , т. е. вычислить произведение $1 \times 2 \times 3 \times \dots \times N$. Тогда можно будет воспользоваться шаблоном команды повторения с параметром и с помощью него составить искомый алгоритм, который можно существенно упростить, если расставить сомножители в обратном порядке, т. е. вычислить произведение $F = N(N-1)(N-2)\dots 1$. В последнем случае можно воспользоваться шаблоном команды повторения по счетчику. Обозначим: $F = N(N-1)(N-2)\dots(N-i)$ — текущее значение произведения, а $K = N-i$ — текущее значение N . Тогда в исходном алгоритме текущее значение произведения будет записано с помощью команды $F = F * K$. Если положить начальное значение $F[0] = 1$, то значение K будет изменяться от N до 1. С учетом сделанных замечаний алгоритм вычисления факториала примет вид алг фактоиал * $N!$ при $N \neq 0$ *

```

    алг RX = N
    рез RX = N!
нач   R0 = K, RY = F = K!
    BB RX = N; НЖ В/О С/П
    K = N
    F = 1
    1 R0 = RX
    2 RX = 1
иц   F = F * K
    3 RX = RX * R0
кц
    до K = 1 от K шаг -1
    4 UNTIL R0 = 1 [FROM R0 STEP -1] GOTO 3
вы F = N!
    5 STOP
кон

```

В этом алгоритме текущее значение $F = K!$ запоминается в регистре Y стека. Оно перемещается в регистр Y после ввода X значения K (БЕЙСИК-команда 3) для вычисления очередного текущего значения F.

6.2.2. КОМАНДА ПМК, ИСПОЛЬЗУЕМАЯ ДЛЯ ПРОГРАММИРОВАНИЯ ПОВТОРЕНИЙ, И ЕЕ ОТображение БЕЙСИК-КОМАНДОЙ

Команда ПМК, реализующая условный переход по счетчику

Для программирования повторений используются все команды условного и безусловного переходов, рассмотренные в п. 6.1. Кроме того, в системе команд ПМК, существенно упрощающих программирование повторений, когда шаг изменения параметра равен -1 , имеется двухшаговая команда условного перехода по счетчику:

$a + 0. FLr$	$a + 1. nn$	$a + 2. h$
1-й шаг	2-й шаг	следующая команда

где $r = 0, 1, 2, 3$ — номер регистра памяти, выделяемый для хранения значения переменной — счетчика обращений к команде; nn — адрес команды, к которой следует перейти, когда содержимое регистра Rr станет равным единице.

Команда FLr выполняется в следующем порядке:

1. Проверяется условие равенства содержимого регистра Rr единице.

2. Если содержимое регистра Rr не равно единице, то из него вычитается единица и осуществляется переход к выполнению команды с адресом nn .

3. Как только содержимое регистра Rr станет равным единице, происходит переход к выполнению следующей команды h с адресом $a + 2$.

Ввод команды условного перехода по счетчику в память ПМК производится путем нажатия клавиши F и одной из клавиш L0, L1, L2, L3, предписывающих хранение значений счетчика в регистрах R0, R1, R2, R3 соответственно, и клавиш nn (двухзначной цифры, указывающей адрес перехода).

Примечание. В инструкциях на все типы ПМК имеется неточность. Вычитание единицы из содержимого счетчика производится не до проверки условия, а после нее. Соответственно переход к выполнению следующей команды происходит не при значении содержимого счетчика, равном нулю, как это сказано в инструкции, а при его значении, равном единице. В этом нетрудно убедиться, прочитав содержимое счетчика с помощью команды PLr после перехода к следующей команде. Однако следует заметить, что в большинстве случаев эта неточность не приводит к ошибкам, так как число повторений и в том и в другом случае остается одинаковым. Ошибка может возникнуть лишь при использовании содержимого счетчика в дальнейших командах программы.

Шаблон БЕЙСИК-команды условного перехода по счетчику

А Б-серия

$B \text{ UNTIL } Rr = 1 [\text{FROM } Rr \text{ STEP } -1] \text{ GOTO } A$

$a+0. П-серия$

$b+0. FLr$

$b+1. a+0$

С следующая Б-команда

$b+2. \text{ следующая П-команда}$

Примечание. Если в БЕЙСИК-команде B вместо A поставить $B+1$, то всегда будет осуществляться безусловный переход к следующей команде, а регистр Rr будет выполнять роль счетчика числа обращений к команде B . Поэтому такая команда может использоваться для организации счетчика.

Пример. Составить программу вычисления факториала в соответствии с алгоритмом, разработанным в предыдущем примере.

ПРОГРАММА вычисления факториала $N!$ при $N \neq 0$

```

алг  факториал
арг R0 = N
рез RX = N!
нач R0 = K, RX = F = K!
    BB RX = N; НЖ В/О С/П;
    K = N
    1 R0 == RX          00. xP0 (R0)
    F = 1
    2 RX = 1            01. 1 (F = 1)
    нц
        F = F * K
    кц
    до K = 1 от K шаг —1
    3 RX = RX * R0      02. пX0 (R0, F)
    03. × (F * R0)
    4 UNTIL R0=1 GOTO 3
        04. FL0
        05. 02
    вы F = N!
    5 STOP              06. С/П (N!)
кон

ИНСТРУКЦИЯ BB RX = N; НЖ В/О С/П; ВЫ N!
ТЕСТ BB R0 = N = 5; НЖ В/О С/П; ВЫ N! = 120 (5 с)
```

Рассмотрим подробно работу составленной программы. Значение N , определяющее число сомножителей факториала и, следовательно, число повторений циклов, вводится с пульта в регистр X до начала работы программы. Это значение автоматически записывается в регистр R0 (БЕЙСИК-команда 1, П-команда 00). Команда 01.1 устанавливает начальное значение факториала, равное единице.

Тело цикла начинается с команды 02.пX0, вызывающей из регистра R0 текущее значение сомножителя факториала K . Одновременно оно указывает, сколько циклов осталось выполнить. При первом обращении к указанной команде $K = N$ и из регистра R0 будет извлечено значение N , которое было занесено туда командой 00. xP0. С каждым новым обращением к этой команде значение K будет уменьшаться на единицу и при последнем обращении станет равным единице. В результате выполнения команды 02 в регистре X будет находиться значение K , а в регистре Y — текущее значение F , полученное перемножением $N(N - 1)(N - 2)\dots 1$.

Команда 03 выполняет умножение текущих значений K и F . При этом значение K находится в регистре X, а F — в регистре Y. Результат умножения помещается в регистр X. После ввода очередного значения (K команда 02) результат $F * K$ перемещается в регистр Y и становится новым значением F .

Команда 04. FL0 05. 02 осуществляет проверку условия $K = 1$. Если оно не соблюдается, то значение K уменьшается на единицу и осуществляется переход к новому повторению — команде 02 с новым значением K . Как только K станет равным единице, произойдет выход из повторения — переход к выполнению команды 06. С/П, предписывающей останов вычислений. При этом текущее значение факториала F станет его окончательным значением и будет отображено на экране индикатора.

6.2.3. ПРИМЕР РЕШЕНИЯ ЗАДАЧИ С ПОВТОРЕНИЯМИ. ПРОГРАММИРОВАНИЕ ФОРМУЛЫ ЭРЛАНГА

Постановка задачи. Дано: Направление связи, состоящее из n телефонных каналов. Коэффициент загрузки направления связи $\alpha = \lambda/\mu$, где λ — интенсивность поступления заявок на переговоры, заявок/ед. времени; μ — интенсивность обслуживания заявок на переговоры, число произведенных переговоров/ед. времени.

Требуется: Вычислить вероятность обслуживания заявки на переговоры.

Решение. Предполагаем, что входящий поток заявок на телефонные переговоры и поток их обслуживания является простейшим, обладающим свойствами:

стационарности (вероятность поступления одинакового числа заявок на одинаковые интервалы времени не зависит от текущего значения времени);

отсутствие последействия (вероятность поступления в заданный интервал времени не зависит от того, сколько заявок уже поступило до этого интервала);

ординарности (вероятность поступления двух и более заявок в один и тот же момент времени пренебрежимо мала).

Для простейшего потока интервала времени между поступлениями соседних заявок и интервалы времени, занимаемые их обслуживанием, являются случайными величинами, подчиненными показательному закону:

для интервалов поступления

$$f_1(t) = \lambda e^{-\lambda t};$$

для интервалов обслуживания

$$f_2(t) = \mu e^{-\mu t}.$$

Тогда вероятность отказа в обслуживании заявки будет определяться по формуле Эрланга

$$P_{\text{отк}} = \frac{\alpha^n}{n!} \left| \sum_{i=0}^{i=n} \frac{\alpha^i}{i!} \right| = \frac{\alpha^n}{n!} \left| \left(1 + \sum_{i=1}^n \frac{\alpha^i}{i!} \right) \right|, \quad (6.1)$$

а вероятность обслуживания телефонных переговоров направлением связи

$$P_{\text{обс}} = 1 - P_{\text{отк.}}$$

Обозначив для кратности $u = \frac{\alpha^n}{n!}$, $v = 1 + \sum_{i=1}^n \frac{\alpha^i}{i!}$, получим $P_{\text{обс}} = 1 - u/v$. Таким образом, расчет $P_{\text{обс}}$ с помощью формулы Эрланга сводится к вычислению величин u и v , которые можно получить, используя программу вычисления факториала и операцию возведения в степень. Однако этот путь в данном случае мало целесообразен, поскольку, с одной стороны, операция возведения в степень выполняется с помощью логарифмирования и среди других операций является самой продолжительной и наименее точной. С другой стороны, вычисление факториала каждый раз заново для каждого слагаемого в знаменателе (без использования ранее полученных результатов) приведет к неоправданно большому объему вычислений. Поэтому вычисление u будем производить по формуле $u = \prod_{i=1}^n \alpha/i$, используя выражение $u_i = u_{i-1} * \alpha/i$, а значение суммы v по формуле $v_i = v_i + u_i$. Тогда вычисление числителя u и знаменателя v может быть осуществлено с помощью команды повторения:

```

u = 1
v = 1
нц
|   u = u * alpha/i
|   v = v + u
кц
до N = 1 от N шаг -1

```

Документация на программу расчета вероятности обслуживания заявки на телефонные переговоры по формуле Эрланга может быть представлена в следующем виде:

ПРОГРАММА 6.1. Расчет вероятности обслуживания заявки на телефонные переговоры:

$$P_{\text{обс}} = 1 - \frac{\alpha N}{N!} \left| \sum_{i=0}^N \frac{\alpha^i}{i!} \right|,$$

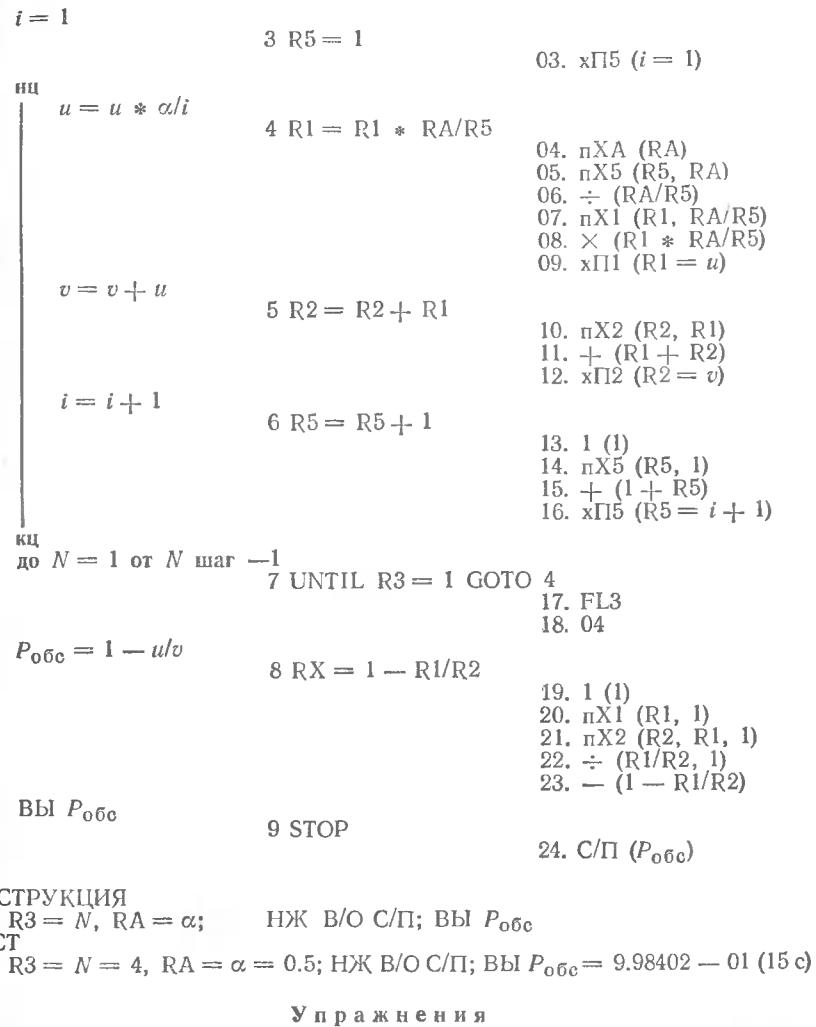
где N — число абсолютно надежных телефонных каналов; α — коэффициент загрузки системы из N каналов; $\alpha = \lambda/\mu$ — интенсивность потока заявок/интенсивности их обслуживания.

алг ЭРЛ
 арг R3 = N, RA = α
 рес RX = $P_{\text{обс}}$
 нач R1 = u , R2 = v , R5 = i
 ВВ R3 = N, RA = α ; НЖ В/О С/П
 u = 1 1 R1 = 1

v = 1

2 R2 = 1

- 00. 1 (1)
- 01. xΠ1 ($u = 1$)
- 02. xΠ2 ($v = 1$)



6.2.1. Разработать программу вычисления среднего значения последовательно вводимой случайной величины $x_1 \dots x_n$. Составить документацию на программу в развернутой форме.

Указание. Программу разработать на основе алгоритма среднее (с. 139).

6.2.2. Разработать программу вычисления суммы членов ряда

$$S = 1 + 1/x + 1/x^2 + 1/x^3 + \dots$$

для $x > 1$ с точностью 0.001. Составить документацию на программу в развернутой форме.

Указание. Программу разработать на основе алгоритма сумма ряда (с. 136).

6.2.3. Разработать программу вычисления суммы N чисел натурального ряда $S = 1 + 2 + 3 + \dots + N$.

6.3. ИСПОЛЬЗОВАНИЕ КОМАНД КОСВЕННОЙ АДРЕСАЦИИ

6.3.1. ПОНЯТИЕ О КОСВЕННОЙ АДРЕСАЦИИ

Обычный способ обращения к регистрам памяти и командам программы путем указания их истинных адресов называется способом прямой адресации. До настоящего времени пользовались этим способом. Однако в ряде задач в момент составления программы истинные адреса некоторых регистров памяти и команд могут быть неизвестны. По условиям задачи их требуется определить (вычислить) уже в самом процессе решения — при выполнении программы. Для вариантов исходных данных эти адреса могут

Таблица 6.2

Команды ПМК с косвенной адресацией

Команды с прямой адресацией		Команды с косвенной адресацией	
Наименование команды	Выполняемые действия	Наименование команды	Выполняемые действия
БП pp	Безусловный переход к выполнению команды с адресом pp	КБП i	Безусловный переход к выполнению команды, адрес которой хранится в регистре i
ПП pp	Переход к выполнению подпрограммы, начальный адрес которой pp	КПП i	Переход к выполнению подпрограммы, начальный адрес которой хранится в регистре i
$Fx = 0 \text{ } nn$ $Fx \neq 0 \text{ } nn$ $Fx \geq 0 \text{ } nn$ $Fx < 0 \text{ } nn$	При несоблюдении условия (x — содержимое регистра X) осуществляется переход к команде с адресом nn , а при соблюдении — к команде, следующей за шагом nn	$Kx = 0i$ $Kx \neq 0i$ $Kx \geq 0i$ $Kx < 0i$	При несоблюдении условия осуществляется переход к команде, адрес которой находится в регистре i , а при соблюдении — к команде, непосредственно следующей за командой с косвенной адресацией
xPi	Запись числа из регистра X в регистр r ($r = 0, 1, 2, \dots, D$)	$KxPi$	Запись числа из регистра X в регистр, номер которого находится в регистре i
Px	Считывание числа из регистра r в регистр X	$KpXi$	Считывание числа из регистра, номер которого находится в регистре i , в регистр X

Примечания: 1. Для всех команд, приведенных в табл. 6.2, для двухшаговых команд $pp = 00, 01, \dots, 97$; для одношаговых команд $r = 0, 1, 2, \dots, D$; $i = 0, 1, 2, \dots, D$.
 Все команды с косвенной адресацией являются одношаговыми.
 2. Для МК-52 и МК-61 $nn = 00, 01, \dots, 97, 98, 99, .0, .1, .2, .3, .4$; $r = 0, 1, 2, \dots, 9$, А, В, С, Д, Е; $i = 0, 1, 2, \dots, 9$, А, В, С, Д, Е.
 3. Содержимое индексного регистра $i = 10$ воспринимается как адрес А; $i = 11$ — В, 12 — С, 13 — Д, 14 — Е.

быть различными. В таких случаях применяют способ косвенной адресации. При этом способе обращение производится путем указания не истинного адреса, а того регистра, в котором хранится вычисленный в ходе выполнения программы истинный адрес. Такой регистр, выделенный для хранения вычисленного значения истинного адреса, называется индексным регистром, а его номер — индексом. Обращение к индексному регистру, выборка значения истинного адреса из него и выполнение команды обращения по адресу, хранящемуся в индексном регистре, осуществляется с помощью специальных команд косвенной адресации. Последние по своей структуре аналогичны командам с прямой адресацией. Различие заключается в том, что все команды с косвенной адресацией являются одношаговыми и начинаются с буквы К (косвенная), в то время как команды с прямой адресацией, включающие в себя обращение к другим командам, являются двухшаговыми. Другое отличие состоит в том, что в конце команды с косвенной адресацией записывается всегда не адрес, а номер индекса из одной цифры. Ввод команд с косвенной адресацией в программную память ПМК производится с использованием префиксной клавиши К. Список команд ПМК с прямой и косвенной адресацией приведен в табл. 6.2.

6.3.2. ОБРАБОТКА ТАБЛИЧНЫХ ДАННЫХ С ПОМОЩЬЮ КОМАНД С КОСВЕННОЙ АДРЕСАЦИЕЙ

В п. 4.1.2 было дано понятие о таблицах и табличных данных, а также отмечалось, что в ПМК практически нет возможности разместить в адресуемой памяти ПМК все элементы таблиц, поскольку ее объем сильно ограничен. Поэтому во всех ранее рассмотренных случаях значения переменных, входящих в состав таблиц, вводились и обрабатывались на ПМК последовательно, с выделением для ввода и хранения всех элементов таблиц только одного регистра памяти. Этот регистр указывался в заголовке исходного алгоритма. Например, таб $RX = A [5]$. Это значит, что для всех пяти элементов таблицы A выделен один регистр памяти RX. Однако довольно часто встречаются задачи, для решения которых необходимо одновременно хранить в адресуемой памяти все элементы таблицы. Если число элементов таблицы небольшое (до 10—12), то решение можно автоматизировать на ПМК, используя команды с косвенной адресацией.

Для простоты здесь и в дальнейшем ограничимся использованием и хранением в памяти ПМК только элементов одномерных таблиц векторов. Поскольку элементы таблицы будут размещаться в адресуемой памяти ПМК, то помимо внешнего имени таблица будет иметь и внутреннее имя, являющееся обозначением регистра, в котором хранится ее первый элемент. Число элементов, подлежащих хранению в адресуемой памяти ПМК (длина таблицы), записывается в квадратных скобках рядом с внутренним именем таблицы, например таб $R2 [5] = A [5]$. Запись в приведенном примере

означает, что для хранения таблицы выделено пять регистров адресуемой памяти, начиная с R2: R2, R3, R4, R5, R6. При записи таб R9 [3] = B [3] будут выделены регистры R9, R4, RB.

Обращение к элементам таблицы, хранящимся в памяти ПМК, в БЕЙСИК-алгоритме осуществляется по ее внутреннему имени и индексу элемента, помещаемому в квадратных скобках рядом с именем таблицы. Например, R2 [R0] означает элемент таблицы R2, индекс которого помещен в регистре R0. Соответственно R9 [R7] — элемент таблицы R9, индекс которого хранится в R7. При этом нумерация индексов начинается с первого элемента таблицы, т. е. индексом первого элемента таблицы всегда будет номер регистра внутреннего имени, например R2 [R0 = 2].

Внешние индексы (переменные) обычно обозначаются буквами i , j , k , и индексные регистры, выделяемые для их хранения, объявляются в заголовке алгоритма после служебного слова нач. Например, нач R0 = i , RA = j . Внешние индексы нумеруются, начиная с нуля или с единицы, как удобнее по условию задачи.

Модификация адреса при выполнении команд с косвенной адресацией. Значения переменных с индексами, обозначающими таблицы, обрабатываются в программе чаще всего последовательно, в порядке возрастания и убывания индекса переменной. Поэтому перед каждым очередным обращением к следующему элементу массива должно быть изменено (или модифицировано) содержимое индексного регистра, чаще всего увеличено или уменьшено на единицу. Такая модификация косвенного адреса (на +1 или -1) может быть выполнена автоматически при каждом очередном обращении к индексному регистру с помощью команды с косвенной адресацией. Шаг изменения адреса (+1 или -1) устанавливается путем соответствующего выбора индексного регистра. Если в качестве индексных регистров используются регистры R0, R1, R2, R3, то при каждом обращении к ним с помощью любой из команд с косвенной адресацией содержимое индексного регистра будет уменьшено на единицу, после чего будет прочитано значение адреса и выполнен переход по модифицированному уже адресу. Если же в качестве индексных регистров будут использованы регистры R4, R5, R6, то перед каждым обращением к ним с помощью команд косвенной адресации их содержимое будет увеличиваться на единицу. Если команды косвенной адресации обращаются к регистрам R7, R8, R9, RA, RB, RC, RD, RE, используемых в качестве индексных, то автоматическая модификация адреса не происходит. После выполнения команд с косвенной адресацией содержимое перечисленных регистров остается без изменений. Для выполнения модификации адреса в случае использования перечисленных регистров в качестве индексных необходимо изменить содержимое индексного регистра с помощью команд программы, помещаемых перед командой с косвенной адресацией.

Обращение к элементам таблицы в случае дробных индексов. В индексные регистры могут записываться не только целые, но и

дробные числа, получающиеся в процессе вычислений. В случае, если содержимое индексного регистра будет представлять собой неправильную десятичную дробь, значение которой больше единицы, то при первом обращении к нему команды с косвенной адресацией это содержимое будет модифицировано путем отбрасывания дробной части. Тогда при последующем считывании содержимого указанного регистра в регистр X в последнем уже будет находиться целое число. Эта особенность выполнения команд с косвенной адресацией используется в МК-54 для выделения целой части числа.

Следует отметить, что в том случае, когда содержимое индексного регистра представляет собой правильную десятичную дробь, т. е. $1 > Ri > 0$, отбрасывание дробной части числа после выполнения команды с косвенной адресацией не всегда имеет место. Поэтому, если предполагается, что содержимое индексного регистра может быть правильной десятичной дробью, т. е. оно меньше единицы, то в программе следует предусмотреть проверку перед выполнением команды с косвенной адресацией. Если содержимое индексного регистра окажется меньше единицы, то следует записать в последний нуль, и лишь после этого использовать команды с косвенной адресацией.

Шаблоны БЕЙСИК-команд считывания и записи переменных с индексами.

1. Шаблон считывания:

A RX = Rr [R1] $a + 0. \text{ КпХ1}$
1 RX = R0 [RA] 00. КпХА

Прочитать элемент таблицы R0 [RA] — это значит прочитать содержимое индексного регистра RA, где записан адрес искомого элемента, и по считанному адресу прочитать в регистре X значение элемента таблицы R0 [RA]. Все это делается с помощью одной одношаговой команды ПМК 00. КпХА.

2. Шаблон записи:

A Rr [R1] = RX $a + 0. \text{ КпП1}$
1 R0 [RA] = RX 00. КпПА

Записать элемент таблицы R0 [RA] из регистра X в адресуемую память — это значит:

прочитать содержимое индексного регистра RA, где хранится индекс элемента таблицы;

по индексу элемента определить регистр, куда следует записать этот элемент;

по определенному регистру записать значение элемента таблицы R0 [RA]. Все эти операции выполняются с помощью одной одношаговой команды ПМК КпПА.

Распределение промежутков времени t поступлений заявок на передачу сообщений между 10-ю частичными интервалами длиной $h = (M - m)/10 = \dots$ и объемом выборки $n = \dots$

Частичный интервал		Частота варианта, попадающих в интервал i , $N[i]$	Плотность частоты $N[i]/h$
Номер	Нижняя граница m [i]		
1	m [1]	N [1]	N [1]/ h
10	m [10]	N [10]	N [10]/ h

П р и м е ч а н и е. В таблице даны абсолютные значения частот и вариант их плотностей. Для получения их относительных значений достаточно разделить абсолютные значения на объем выборки n .

1-я часть (подготовительная) — вычисление длины интервала и обнуление всех регистров, в которых будут храниться элементы массива, состоящие из 10 счетчиков частот, вариант, попадающих в соответствующие частичные интервалы. Выполняемые действия:

1. Ввод границ диапазона изменения случайной величины t : M (верхняя граница) и m (нижняя граница).
2. Вычисление и выдача длины частичного интервала h .
3. Обнуление всех 10 регистров массива N (массива счетчиков частот). Тем самым задается начальное состояние каждого интервального счетчика, равное нулю.

2-я часть — ввод очередной случайной величины t , определение частичного интервала, в который она попадает, и увеличение на единицу содержимого счетчика частот соответствующего интервала. Выполняемые действия:

1. Ввод очередного случайного числа $t[j]$. Введенное число записывается в регистр R0, где была ранее записана верхняя граница M диапазона изменения случайной величины. Значение $t[j]$ остается в регистре R0 до окончания процедуры его обработки, т. е. до ввода случайного числа $t[j+1]$.

2. Определение номера частичного интервала i , в который попадает введенная случайная величина $t[j]$:

$$i = (t[j] - m)/h + 1.$$

Значение i при этом получится в виде десятичной дроби, которая в общем случае может быть положительной, отрицательной, а также равной нулю.

3. Проверка, попадает ли число i в допустимые значения номеров интервалов $1 \leq i \leq 10$. Если для введенного значения $t[j]$ вы-

Постановка задачи. Да и о: В экспедицию узла связи поступает поток заявок на передачу сообщения. Промежутки времени t между поступлениями предыдущей и последующей заявок составляют $t = t_1, t_2, t_3, \dots, t_j \dots$.

Требуется: Разработать документацию на программу расчета таблицы для построения гистограммы распределения промежутков времени между поступлениями заявок на передачу сообщений.

Решение. Пусть величина t принимает значения от минимального $t = m$ до максимального $t = M$. Вне этих пределов значение t нас не интересует, такие значения можно рассматривать, как выбросы.

Диапазон изменения случайной величины t , равный $M - m$, разбьем на некоторое конечное число одинаковых интервалов длиною h каждый, которые будем называть частичными интервалами. Учитывая ограниченные возможности ПМК (недостаточное число регистров оперативной памяти), количество интервалов установим постоянным и равным 10. Тогда длина одного интервала $h = \frac{M - m}{10}$ и нижние границы между частичными интервалами будут определяться так:

$$\begin{aligned}m &[1] = m + 0 * h; \\m &[2] = m + 1 * h; \\m &[i] = m + (i - 1) * h; \\m &[10] = m + 9 * h.\end{aligned}$$

Наблюдаемое значение случайной величины t принято называть ее вариантом или вариантой, а число наблюдаемых значений $N[i]$ — частотой. Отношение частоты к длине интервала $N[i]/h$ принято называть плотностью частоты.

Нашей задачей будет определить границы частичных интервалов, а также частоты и плотности частот вариантов, попадающих в каждый частичный интервал. Достаточно определить лишь нижние границы частичных интервалов. Верхние границы при этом определяются автоматически: нижняя граница следующего интервала будет верхней границей текущего интервала. Результаты расчета представим в виде табл. 6.3.

Помещаемые в таблицу результаты расчета предназначены для построения гистограмм распределения случайной величины t . Пример построения гистограммы по результатам обработки наблюдений, помещаемых в табл. 6.3, будет рассмотрен ниже.

Алгоритм расчета, приведенного в табл. 6.3, можно подразделить на три составные части.

численное значение i выйдет за указанные пределы, то $t [j]$ рассматривается как выброс и не учитывается при подсчете числа введенных величин $t [j]$. Если же число $t [j]$ попадает в допустимые границы диапазона изменений m и M или, что то же, в численное значение i попадает в предел допустимых номеров интервалов (1 — 10), то можно утверждать, что число $t [j]$ попадает в интервал $mh [i] \leq t < mh [i + 1]$, где $mh [i]$ и $mh [i + 1]$ — нижняя и верхняя границы интервала.

Рассмотрим на примерах, как вычисляется номер частичного интервала. Пусть $m = 10$, $M = 60$. Тогда $h = (60 - 10)/10 = 5$ и для различных значений случайной величины t соответствующие значения i будут следующими:

t	10	12	20	40	59	60
i	1	1	3	7	10	11

4. Увеличение на единицу содержимого соответствующего интервального счетчика. Для создания 10 интервальных счетчиков образуется таблица, которая описывается в заголовке алгоритма, как таб R1 [10] = $N [10]$. Таблица N состоит из 10 элементов — переменных с индексами, для хранения которых выделяется 10 регистров адресуемой памяти с адресами от I до A. Для хранения значений индекса выделен регистр D. Обращение к элементу массива $N [i]$ (интервальному счетчику) производится с помощью команды с косвенной адресацией KпXD. Эта команда вызывает из регистра D целую часть его содержимого, равную значению индекса i , являющегося адресом элемента таблицы. По этому адресу находится соответствующий регистр адресуемой памяти и из последнего считывается число в регистр X. Это число представляет собой показание интервального счетчика. Затем к считанному числу добавляется единица и результат помещается в регистр X. Далее с помощью команды с косвенной адресацией KпID полученнное новое значение интервального счетчика засыпается в тот регистр, откуда было прочитано его предыдущее значение.

5. Возврат к началу второй части программы и останов для ввода следующего случайного числа $t [j + 1]$. При этом на экране индикатора отображается предыдущее, ранее введенное число $t [j]$. Если оно выходит за пределы допустимого диапазона, то его отображение имеет знак минус. В противном случае слева отображается порядковый номер числа j , а справа — значение числа $t [j]$. Формирование отображения на экране, включающего искомое число и его порядковый номер, осуществляется с помощью команды знач = $j * 10^5 + t [j]$. Если $j = 3$, $t [j] = 15$, то на экране индикатора будет отображено 300015.

З-я часть — выдача результатов решения в форме, наиболее приемлемой для заполнения бланка табл. 6.3. Осуществляется по всем десяти интервалам в следующей последовательности:

номер интервала i ;

нижняя его граница $mh [i]$;
частота попадания в интервал $N [i]$;
плотность частоты $N [i] h$.

После этого на экране индикатора должен быть отображен признак окончания решения — число 777.

3-я часть является обслуживающей, обеспечивающей выдачу результатов в последовательности, наиболее удобной для заполнения бланка табл. 6.3. Если необходимо знать только содержимое интервальных счетчиков, то выполнение третьей части можно опустить, прочитав содержимое интервальных счетчиков с помощью ручного набора команд pX1, pX2, pX3, ..., pX9, pXA.

С учетом изложенного алгоритм расчета таблицы для построения гистограммы распределения случайных величин t будет выглядеть следующим образом:

ПРОГРАММА 6.2. Расчет таблицы для построения гистограммы распределения случайных чисел $t = t [1], t [2], t [3], \dots, t [j], \dots, t [n]$. Диапазон изменения t : m — нижняя граница, M — верхняя граница. Число частичных интервалов — 10.

Исходные данные: $M, m, t = t [1], t [2], \dots, t [j], \dots, t [n]$.

Результаты:

длина частичного интервала $h = (M - m)/10$;
порядковый номер интервала $i = 1, 2, 3, \dots, 9, A$;
нижняя граница интервала $m, mh [1], mh [2], \dots, mh [10]$;
частота попадания в i -й интервал $N [1], N [2], \dots, N [10]$;
плотность частоты попадания в i -й интервал $N [1]/h, N [2]/h, \dots, N [10]/h$;
объем выборки n — количество случайных чисел t , попадающих в заданный диапазон изменения (m, M).

Ограничения: $j \geq 1, M > m, n$ — неограничено.

алг **гистограмма**
арг $R0 = M, RB = m$, таб $RX = t$
рез $RD = i, RE = n, RC = h$, таб $RX = mh, R1 [10] = N [10]$

нач $RE = j, \text{таб } R0 = t$
* 1-я часть. Обнуление интервальных счетчиков. Определение h *

$BB R0 = M, RB = m$

НЖ В/О С/П

$i = 0$

1 $RD = 0$

00. Cx (0)
01. xPID ($RD = 0$)

иц $i = i + 1$

2 $RD = RD + 1$

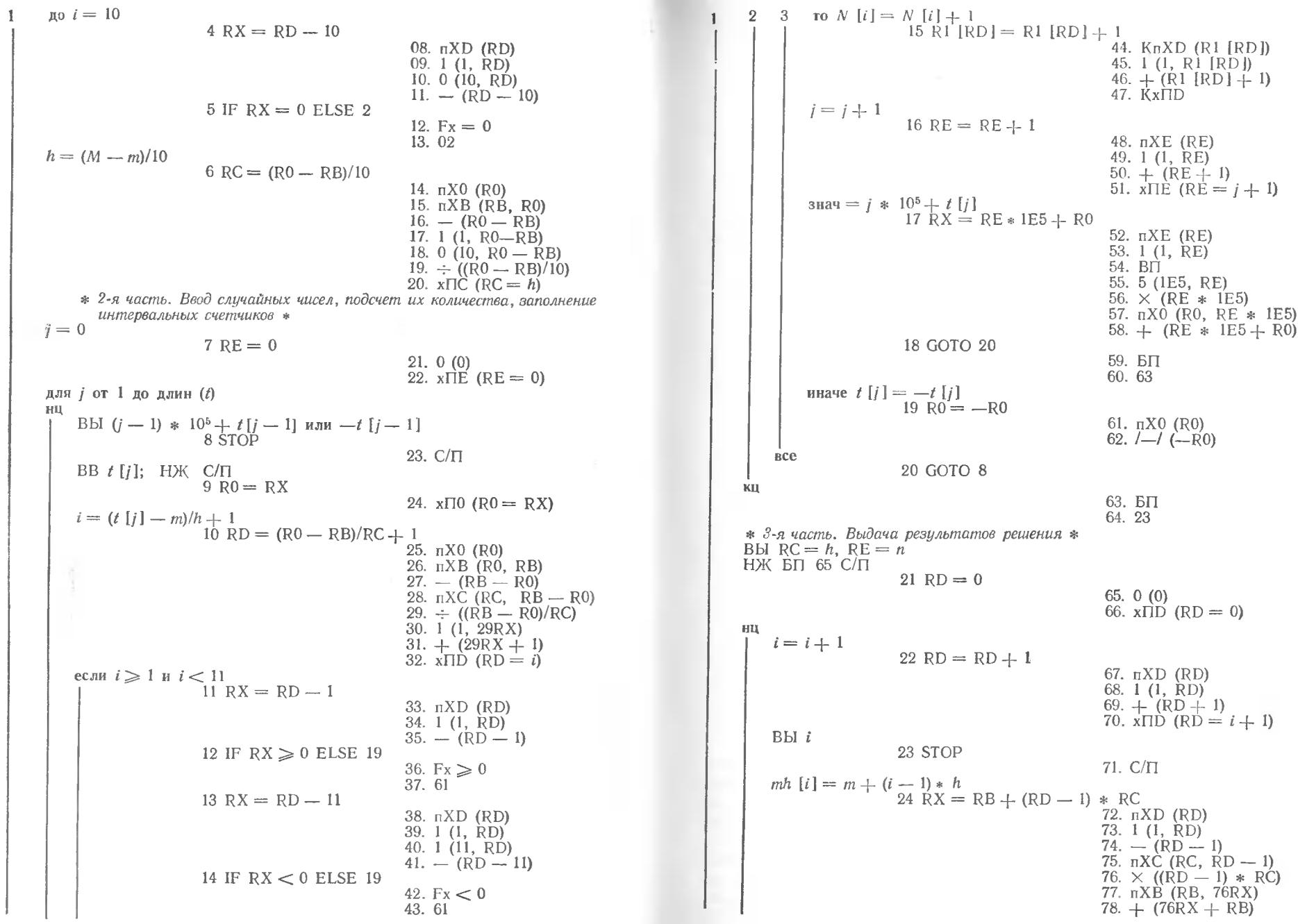
02. пXD (RD)
03. 1 (1, RD)
04. + ($RD + 1$)
05. xPID ($RD = i + 1$)

$N [i] = 0$

3 $R1 (RD) = 0$

06. 0 (0)
07. KпID ($N [i] = 0$)

кц



ВЫ $mh[i]$ 25 STOP
 ВЫ $N[i]$ 26 RX = R1 [RD]
 ВЫ $N[i]$ 27 STOP
 ВЫ $N[i]/h$ 28 RX = RX/RC
 кц до $i = 10$ 29 STOP
 30 RX = RD - 10
 31 IF RX = 10 ELSE 22
 ВЫ 777 32 RX = 777
 кон 33 STOP
 ИНСТРУКЦИЯ
 * 1-я часть. Обнуление счетчиков.*
 ВВ R0 = M, RB = m; НЖ В/О С/П; ВЫ h
 * 2-я часть. Ввод $t[j]$ и заполнение счетчиков $N[i]$
 для j от 1 до длин (t)
 иц
 | если $t[j-1] \geq m$ и $t[j-1] < M$
 | то ВЫ $(j-1) * 10^5 + t[j-1]$
 | иначе ВЫ $-t[j-1]$
 все
 ВВ $t[j]$
 НЖ С/П
 кц
 * 3-я часть. Выдача результатов. *
 ВЫ RC = h, R0 = n
 НЖ БП 65 С/П
 для i от 1 до 10
 иц
 | ВЫ i ; НЖ С/П
 | ВЫ $mh[i]$; НЖ С/П
 | ВЫ $N[i]$; НЖ С/П
 | ВЫ $N[i]/h$; НЖ С/П
 кц
 ВЫ 777
 ТЕСТ
 * 1-я часть. Обнуление счетчиков. *
 ВВ R0 = M = 60, RB = m = 10; НЖ В/О С/П ВЫ 0 (40 с)

* 2-я часть. Ввод $t[j]$ и заполнение счетчиков $N[i]$ *
 для j от 1 до длин (t)
 иц

j	ВВ $t[j]$	НЖ С/П ВЫ $t[j]$	j	ВВ $t[j]$	НЖ С/П ВЫ $t[j]$
1	7	-7(15с)	7	43	700043
2	10	100010	8	33	800033
3	12	200012	9	48	900048
4	23	300023	10	53	1000053
5	17	400017	11	58	1100058
6	38	500038	12	59	1200059
	28	600028	—	60	-60

кц
 3-я часть. Выдача результатов.*

НЖ БП 65
 для i от 1 до 10
 иц

НЖ С/П ВЫ i	НЖ С/П ВЫ $mh[i]$	НЖ С/П ВЫ $N[i]$	НЖ С/П ВЫ $N[i]/h$
1	10	2	4 -01
2	15	1	2 -01
3	20	1	2 -01
4	25	1	2 -01
5	30	1	2 -01
6	35	1	2 -01
7	40	1	2 -01
8	45	1	2 -01
9	50	1	2 -01
10	55	2	4 -01

кц
 НЖ С/П; ВЫ 777, RC = h = 5

Некоторые особенности алгоритма
 и программы расчета таблицы для построения гистограммы

1. В связи с тем, что 10 регистров памяти ПМК заняты хранением значений интервальных счетчиков, для хранения значений других переменных остается мало регистров. Поэтому оставшиеся регистры многократно используются для хранения значений различных переменных.

2. Применяемый способ индикации порядкового номера случайной величины j и ее значения $t[j]$ ограничивает длину последовательности длин (t) значением $j_{max} = 999$. Однако, уменьшив значение множителя, формирующего отображение на экране индикатора с 10^5 до 10^4 , можно увеличить диапазон изменения j на порядок.

Распределение промежутков времени между моментами поступления заявок на передачу сообщений между десятью частичными интервалами ($h = (M - m) / 10 = 3$, $M = 30$ мин, $m = 0$ мин, объем выборки $n = 100$)

Частичный интервал		Частота попаданий в интервал	Плотность частоты
номер	нижняя граница		
1	0	30	10
2	3	20	6.6666666
3	6	15	5
4	9	12	4
5	12	9	3
6	15	7	2.3333333
7	18	3	1
8	21	2	6.6666666-01
9	24	1	3.3333333-01
10	27	1	3.3333333-01

Построение гистограммы. Напомним, что гистограммой называют ступенчатую фигуру, состоящую из прямоугольников, основаниями которых служат частичные интервалы h , а высоты равны отношениям $N [i] / h$ (плотностям частот). Для построения гистограммы на оси частот откладывают частичные интервалы h , а над ними проводят отрезки, параллельные оси абсцисс на расстоянии $N [i] / h$.

Площадь i -го прямоугольника равна $h * N [i] / h = N [i]$, т. е. частоте попадания в i -й интервал. Площадь всей гистограммы равна сумме всех частот $N [i]$, т. е. объему выборки n . Гистограмма, построенная на основе табл. 6.4, представлена на рис. 6.1. Из нее видно, что закон распределения промежутков времени между моментами поступления заявок на передачу сообщений близок к экспоненциальному. В принципе использованная нами программа 6.2 пригодна для обработки самых различных случайных величин и построения соответствующих гистограмм.

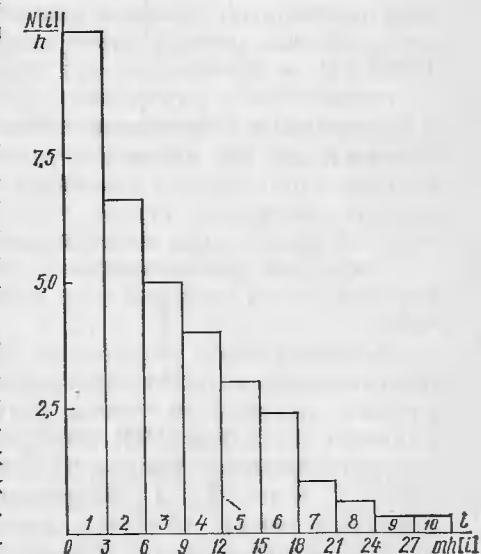


Рис. 6.1. Гистограмма распределения промежутков времени между моментами поступления заявок на передачу сообщений. Объем выборки $n = 100$

3. В рассматриваемой программе использованы все регистры адресуемой памяти МК-61 и МК-52, поэтому ее выполнение на МК-54 в полном объеме не представляется возможным. Однако если отказаться от автоматического подсчета числа вводимых случайных величин, упростив соответственно алгоритм и программу, то задачу можно решить и на МК-54. Для этого из приведенного алгоритма и программы необходимо исключить БЕЙСИК-команды 7, 15, 16 и детализирующие их команды ПМК.

4. Разработанная программа имеет длину 94 шага и занимает почти всю программную память ПМК. Усложнение программы обусловлено наличием в ней развитой обслуживающей сервисной части, облегчающей ввод исходных данных и использование выдаваемых результатов. Эта часть составляет почти половину длины программы. Такое усложнение вполне оправдано, так как существенно упрощает выполнение вычислений по программе. Однако разработка таких программ вызывает определенные трудности, в связи с чем более целесообразно производить ее по частям. В программе имеются четко разделяемые функциональные части: ввод очередного случайного числа; заполнение интервальных счетчиков; формирование отображения на экране индикатора номера случайного числа и его значения. Перечисленные части можно разработать отдельно и представить в виде подпрограмм. Тогда основная программа будет содержать лишь ссылки на подпрограммы. При этом основная программа станет более обозримой и компактной. Способы разработки подпрограмм и обращения к ним в основных программах будут подробно рассмотрены в следующей главе.

6.3.4. РЕШЕНИЕ ЗАДАЧИ ПО РАЗРАБОТАННОЙ ПРОГРАММЕ

Постановка задачи. Дано: 100 значений промежутков времени в минутах между моментами поступления заявок на передачу сообщений:

1	2	1	2	0	0	1	2	0	1	3	4	5	3	5	4	6	8	7	3
6	7	9	10	11	9	13	14	12	15	16	15	18	20	23	21	25	29	0	1
2	1	2	1	0	1	2	2	4	5	3	5	4	7	8	7	6	3	4	6
10	9	11	10	13	12	14	16	15	19	2	1	2	1	0	2	0	1	0	1
3	14	8	12	6	13	7	17	8	16	5	6	3	11	4	9	5	10	4	11

Диапазон изменений величины t : $t = 0 \div 30$ мин.

Требуется: Используя программу 6.2, рассчитать таблицу распределения промежутков времени между моментами поступления заявок на передачу сообщений и по данным таблицы построить гистограмму распределения случайной величины t .

Решение. Задача решается в следующем порядке:

1. В программную память ПМК вводится программа решения задачи.
2. По тесту проверяется правильность ее ввода и работоспособность программы.

3. В точном соответствии с инструкцией выполняется решение, в процессе которого осуществляется ввод исходных данных, запуск программы и считывание результатов с экрана индикатора. Полученные данные заносятся в табл. 6.4.

Упражнение

6.3.1. Используя программу 6.2, установить, сколько раз выпадет грань игральной кости при многократном ее бросании.

6.4. ОСОБЕННОСТИ РАЗРАБОТКИ И ОТЛАДКИ СЛОЖНЫХ ПРОГРАММ С ВЕТВЛЕНИЯМИ И ПОВТОРЕНИЯМИ

6.4.1. ВВОД И ВЫВОД ДАННЫХ В СЛОЖНЫХ ПРОГРАММАХ

Применительно к ПМК используются два способа ввода исходных данных и вывода результатов: адресный и безадресный. Оба способа использовались нами ранее в рассмотренных выше программах.

Адресный ввод-вывод. Адресный ввод исходных данных производится до начала работы программы, перед ее запуском. Он характеризуется тем, что исходные данные непосредственно перед запуском программы записываются вручную в выделенные для них регистры адресуемой памяти ПМК. Адресный ввод предписывается с помощью команды алгоритмического языка ВВЕСТИ *регистр 1 = имя 1 [, регистр 2 = имя 2 ...]*.

Например, ВВЕСТИ R0 = ψ , R1 = n (или ВВ R0 = ψ , R1 = n).

Преимуществами адресного ввода являются:

сокращение длины программы;

простое решение вопроса с идентификацией значений вводимых переменных. Значение каждой переменной вводится во вполне определенный регистр адресуемой памяти, указанный в команде ВВЕСТИ, и приобретает при этом свое внутреннее имя;

существенное упрощение повторного выполнения программы с измененными значениями исходных данных. В этом случае вводу подлежат не все значения переменных, а только те, значения которых претерпевают изменения при повторном выполнении (при расчёте следующей строки таблицы обычно требуется дополнительный ввод только одной переменной).

Последнее преимущество является решающим. Поэтому в тех случаях, когда адресный ввод возможен, его используют наиболее часто.

Адресный вывод результатов обычно производится после полного окончания работы программы и заключается в выполнении ручных операций по чтению результатов из отведенных для их хранения регистров ПМК. Он предписывается с помощью команды алгоритмического языка ВЫДАТЬ *регистр 1 = имя 1 [, регистр 2 = имя 3 ...]*. Например, ВЫДАТЬ R0 = p , R1 = K . Адресный вывод обладает двумя преимуществами: простотой идентификации значений результатов по именам внутренних переменных и сокращением длины программы за счет исключения команд записи результатов в регистр X и остановов для их считывания с экрана индикатора. Однако решающее преимущество адресного ввода при адресном выводе отсутствует. Даже при из-

менении только одного исходного данного при вводе необходимо считывать из регистров памяти все результаты вручную. Поэтому адресный вывод применяется обычно в тех случаях, когда возникают трудности с идентификацией большого количества результатов или размещением программы в памяти ПМК.

Безадресный ввод-вывод. Кроме случаев, когда адресный ввод становится невозможным из-за недостаточного объема адресуемой памяти ПМК, он может оказаться и нецелесообразным в силу присущего ему недостатка — большого количества ручных операций при выполнении ввода-вывода. Поэтому часто используют безадресный ввод исходных данных, а также безадресный вывод результатов. В отличие от адресного ввода-вывода в безадресном исходные данные при вводе записываются в регистры стековой памяти, а результаты считаются из них. При этом ввод и вывод каждого очередного значения осуществляются через один и тот же регистр X. Перемещение введенных значений из регистра X в другие регистры стека происходит автоматически. Перемещение результатов в регистр X для их чтения осуществляется вручную с помощью команд ПМК F. и \leftrightarrow , причем последняя команда используется для чтения содержимого регистра Y. Безадресный ввод предписывается с помощью команд алгоритмического языка:

$$\left\{ \begin{array}{l} \text{ВВЕСТИ} \\ \text{ВВ} \end{array} \right\} \left[\begin{array}{l} \left\{ \begin{array}{l} \text{RX} \\ \text{RY} \\ \text{RZ} \\ \text{RT} \end{array} \right\} = \text{имя 1} \\ , \left\{ \begin{array}{l} \text{RX} \\ \text{RY} \\ \text{RZ} \\ \text{RT} \end{array} \right\} = \text{имя 2} \dots \end{array} \right]$$

или

$$\left\{ \begin{array}{l} \text{ВВЕСТИ} \\ \text{ВВ} \end{array} \right\} = \text{имя 1} [! \text{ имя 2 } \dots]$$

При этом значение последней переменной справа вводится в регистр X, предпоследнее в регистр Y и т. д. (не более четырех переменных!).

При м е р ы:

1. ВВЕСТИ RX = n , RY = m , ВВЕСТИ $n ! m (m, n)$, ВВ h.

Безадресный вывод предписывается с помощью команды:

$$\left\{ \begin{array}{l} \text{ВЫДАТЬ} \\ \text{ВЫ} \end{array} \right\} \left[\begin{array}{l} \left\{ \begin{array}{l} \text{RX} \\ \text{RY} \\ \text{RZ} \\ \text{RT} \end{array} \right\} = \text{имя 1} \\ , \left\{ \begin{array}{l} \text{RX} \\ \text{RY} \\ \text{RZ} \\ \text{RT} \end{array} \right\} = \text{имя 2} \dots \end{array} \right]$$

или

ВЫДАТЬ имя 1 [, имя 2 ...]

2. ВЫ RX = n , RY = m , RZ = K ; ВЫДАТЬ $n, m K$.

После выполнения команд ВВЕСТИ и ВЫДАТЬ для продолжения выполнения программы необходимо нажать клавишу С/П. При этом осуществляется пуск программы с адреса, на единицу большего, чем адрес предшествующей команды останова.

Идентификация исходных данных и результатов при безадресном вводе-выводе. При большом количестве вводимых исходных данных и выводимых результатов и использовании безадресного ввода-вывода возникает необходимость в их идентификации (чтобы

не перепутать при вводе или выводе). С этой целью вводимые и выводимые данные снабжаются специальными признаками, отображаемыми на экране индикатора. Отображение на экране индикатора признака исходного данного осуществляется непосредственно перед его вводом и служит для пользователя предписанием выполнить указанный ввод. Признак результата отображается на экране индикатора либо одновременно с результатом, либо отдельно от него в регистре X. При этом результат помещается в регистре Y и считывается с экрана после нажатия клавиши \leftrightarrow . Признак данного должен существенно отличаться от его значения. Все признаки формируются с помощью одной или нескольких команд программы. В качестве таких признаков используются:

1. Порядковые номера вводимых исходных данных. Для нумерации исходных данных в программе организуется счетчик вводимых данных, который в начале программы обнуляется. Перед началом ввода на экране индикатора отображается порядковый номер либо предыдущего, введенного уже данного, либо очередного, которое предполагается ввести. При вводе очередного данного признак (номер) перемещается в регистр Y.

2. Ранее введенное предыдущее значение исходного данного. Указанное значение должно быть сохранено в памяти ПМК до момента ввода следующего значения и отображено на экране перед вводом очередного в качестве его признака. После ввода очередного данного предыдущее перемещается в регистр Y.

3. Предыдущее значение исходного данного и одновременно его порядковый номер. В этом случае перед вводом в одной части экрана отображается предыдущее значение, а в другой — порядковый номер. Например,

1200327
 $\overbrace{\quad}^{\uparrow}$
 $\overbrace{\quad}^{\uparrow}$
 |
 Значение данного
 Порядковый номер данного

Указанный способ идентификации данных был применен в рассмотренной ранее программе 6.2 (расчет таблицы для построения гистограммы). Здесь, как и в предыдущем случае, организуется счетчик вводимых величин, который в начале программы обнуляется. Перед вводом очередного исходного данного формируется число, содержащее информацию о значении и порядковом номере ранее введенного числа. Это делается следующим образом. Пусть $t[i]$ — последнее введенное значение переменной t , а i — его порядковый номер. Тогда отображаемое на экране индикатора число формируется так: $i * 10^n + t[i]$, где $n = 7, 6, 5 \dots$ соответственно для однозначного, двузначного, трехзначного максимального значения i . Тогда в левой части экрана будет отображено значение i , а в правой — значение $t[i]$, соответствующее предыдущему вводу.

Рекомендуемые признаки идентификации исходных данных и результатов при безадресном вводе-выводе

Признак	Способ получения	
	Команды БЕЙСИК-ПМК	Команды программы
3.1415926 1.1111111—01	RX = P1 RX = 1/9	$a + 0. F\pi (\pi)$ $a + 0.9 (9)$ $a + 1. Fl/x (1/9)$
3.3333333—01	RX = 1/3	$a + 0.3 (3)$
5.5555555—01	RX = 1/1.8	$a + 1. Fl/x (1/3)$ $a + 0.1$ $a + 1.$ $a + 2.8 (1.8)$ $a + 3. Fl/x (1/1.8)$
6.6666666—01	RX = 1/1.5	$a + 0.1$ $a + 1.$ $a + 2.5 (1.5)$ $a + 3. Fl/x (1/1.5)$
ЕГГОГ	RX = ЕГГОГ	$a + 0. K1/x (ЕГГОГ)$ или $a + 0. K\sqrt{-}$ или $a + 0. Kx^2$
8.Е	RX = NOT (11)	$a + 0.1$ $a + 1. 1 (11)$ $a + 2. КИНВ (8.Е)$
8.Г	RX = NOT (12)	$a + 0.1$ $a + 1. 2 (12)$ $a + 2. КИНВ (8.Г)$
8.І	RX = NOT (13)	$a + 0.1$ $a + 1. 3 (13)$ $a + 2. КИНВ (8.)$
8.Л	RX = NOT (14)	$a + 0.1$ $a + 1. 4 (14)$ $a + 2. КИНВ (8.)$
8.	RX = NOT (15)	$a + 0.1$ $a + 1. 5 (15)$ $a + 2. КИНВ (8.—)$

Примечания: 1. Функция NOT (...) в БЕЙСИК-ПМК и соответствующая ей команда программы КИНВ предполагают выполнение инверсии (логическое отрицание) аргумента, помещенного в регистр X. При этом команда КИНВ на первую цифру числа не реагирует (она должна быть обязательно значащей). Вместо первой цифры в результате всегда помещается цифра 8 с точкой. Остальные цифры числа, помещенного в регистр X, заменяются следующими символами:

Цифра исходного числа в регистре	Результат выполнения КИНВ	Цифра исходного числа в регистре Х	Результат выполнения КИНВ
0	—	5	—
1	Е	6	9
2	Г	7	8
3	—	8	7
4	Л	9	6

2. Признаки 7—11 могут быть использованы только на МК-52 и МК-61, где предусмотрены команды КИНВ. При этом количество символов в признаком может быть расширено путем увеличения цифр исходного числа.

В ряде случаев таким же образом можно идентифицировать и значения выводимых результатов. В левой части — порядковый номер выводимого результата (или номер графы таблицы, куда следует записать результат), а в правой части — искомое значение (результат).

4. Специальные признаки, легко отличимые от вводимых данных и результатов решения. В качестве таких признаков можно использовать характерные числа, например, состоящие из одних единиц, двоек, троек, семерок и т. д., появление которых в качестве результатов маловероятно. Могут быть также использованы комбинации десятичных цифр и символов: L [Г Е, отображаемых на экране индикатора. Перечень рекомендуемых признаков и способы их формирования приведены в табл. 6.5.

6.4.2. ОСОБЕННОСТИ ОТЛАДКИ СЛОЖНЫХ ПРОГРАММ С ВЕТВЛЕНИЯМИ И ПОВТОРЕНИЯМИ

Все процедуры отладки программ, изложенные в п. 5.4, пригодны также и для сложных программ, включающих ветвления и повторения. Однако отладка последних имеет свои особенности, а именно:

1. При выполнении программы в режиме пошаговой проверки одно нажатие клавиши ПП обеспечивает выполнение только одной команды, независимо от того, одношаговая она или двухшаговая. Для двухшаговых команд, к которым относятся все команды условных и безусловных переходов, однократное нажатие клавиши ПП вызывает выполнение сразу двух шагов программы, образующих соответствующую двухшаговую команду. При этом содержимое счетчика адреса команд увеличивается сразу на две единицы.

2. Для исправления команд условных и безусловных переходов необходимо в режиме АВТ набрать команду БП_{nn}, где nn — адрес первого шага двухшаговой команды, которую надлежит исправить. При этом в программную память вновь должны быть записаны оба шага исправляемой команды, независимо от того, какой из шагов следует исправить.

3. При выполнении программ в режиме пошаговой проверки после выполнения каждой команды перехода необходимо проанализировать, по какому адресу фактически передано управление. Для этого необходимо перевести ПМК из режима АВТ в режим ввода программы, нажав клавиши ФПРГ. Тогда в правой части индикатора будет отображен искомый адрес — адрес той команды, которая будет выполнена при очередном нажатии клавиши ПП. Если передача управления произведена правильно, то продолжить пошаговую проверку, нажав предварительно клавиши FABT.

4. При аварийном останове программы с отображением на индикаторе сигнала ЕГГОГ необходимо определить адрес останова, нажав для этого клавиши ФПРГ. Для продолжения решения необходимо снова перевести ПМК в режим вычислений, нажав клавиши FABT.

5. В процессе отладки должны быть проверены все ветви программы. Повторения должны быть проверены не менее чем на трех циклах.

6. В процессе отладки программ повышенной сложности имеют место случаи, когда программа оказывается вполне работоспособной в режиме пошаговой проверки, но не работает в автоматическом режиме. Это проявляется либо в искажении исходных данных, либо в зацикливании программы, т. е. в появлении паразитных передач управления командами, в которых эти передачи не предусмотрены программой. Наиболее часто это проявляется после выполнения команд безадресного ввода. Для того чтобы заставить ПМК работать в автоматическом режиме, обычно достаточно отделить последнюю часть программы (которая не выполняется, от той, в которой возникает сбой) командой КНОП, не выполняющей никакой операции. В случае безадресного ввода команда КНОП помещается после останова вычислений в качестве первой команды программы ее продолжения.

Раздел IV

РАЗРАБОТКА ПРОГРАММНЫХ КОМПЛЕКСОВ

Глава 7. ОБЩИЙ ПОРЯДОК РАЗРАБОТКИ ПРОГРАММНЫХ КОМПЛЕКСОВ

В процессе работы с ПМК у пользователя постоянно накапливается личная библиотека программ, неоднократно использованных при решении различных задач. Естественно, возникает потребность в применении имеющегося задела при разработке новых программ для ПМК путем включения их во вновь разработанные (целиком или отдельными частями). Это требует составления каждого нового алгоритма и программы таким образом, чтобы максимально облегчить их включение в алгоритмы и программы, разрабатываемые в дальнейшем. Поэтому на всех уровнях представления алгоритма вводится еще одна команда обращения к вспомогательному алгоритму (программе). Она может рассматриваться как укрупненная команда (макрокоманда), выполнение которой осуществляется с помощью вспомогательной программы, называемой *подпрограммой*. После выполнения последней должен быть обеспечен автоматический возврат к дальнейшему выполнению основной программы.

При достаточно большом количестве алгоритмов и программ в личной библиотеке пользователя (включая вспомогательные алгоритмы и программы) создание новых программ будет сведено в значительной мере к их сборке в программные комплексы. В данной главе будут рассмотрены вопросы, касающиеся построения вспомогательных алгоритмов и подпрограмм, их включения в основной алгоритм и программу, а также сборки имеющихся программ и подпрограмм в программные комплексы.

7.1. ВСПОМОГАТЕЛЬНЫЙ АЛГОРИТМ

7.1.1. ОСОБЕННОСТИ ВЫПОЛНЕНИЯ ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА

В качестве вспомогательного алгоритма может выступать, вообще говоря, любой алгоритм, оформленный в соответствии с требованиями, изложенными в п. 5.2. Вспомогательный алгоритм, как и основной, включает в себя все три уровня его представления: исходный алгоритм пользователя, БЕЙСИК-алгоритм и программу в виде последовательности команд ПМК. Детализация команд старшего уровня командами младшего уровня осуществляется путем использования шаблонов, ранее рассмотренных, и тех, которые будут введены.

Основной алгоритм лишь в одном существенно отличается от вспомогательного: в конце последнего должна быть помещена

команда, обеспечивающая возврат к выполнению основного алгоритма, шаблон которой имеет вид:

ВОЗВРАТ

A + 0 RETURN

a + 0. В/О

Команда ВОЗВРАТ обеспечивает переход к выполнению той команды исходного алгоритма, которая записывается вслед за командой обращения к вспомогательному алгоритму. БЕЙСИК-команда RETURN является аналогом команды ВОЗВРАТ исходного алгоритма. Она обеспечивает возврат к выполнению команды основного БЕЙСИК-алгоритма, непосредственно следующей за командой обращения к вспомогательному. Команда ПМК В/О (возврат обратно) является аналогом БЕЙСИК-команды RETURN. Она обеспечивает возврат к выполнению команды основной программы, помещенной вслед за командой обращения к подпрограмме (после выполнения последней).

П р и м е ч а н и е. Команда подпрограммы В/О предписывает возврат из подпрограммы. В ручном режиме нажатие клавиши В/О предписывает сброс счетчика адреса команд на нуль, т. е. установку нулевого пускового адреса.

7.1.2. ВЫЗОВ ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА

Команда вызова имеет следующий вид:

имя алгоритма (арг регистр 1 = имя [, регистр 2 = имя ...],
рез регистр 3 = имя [, регистр 4 = имя ...],
рг список рабочих регистров)
БРВ (арг R1 = n, R2 = m, R6 = K, рез = K_р rg R7, R8)

В отличие от алгоритмического языка (лексикона) [14] в предлагаемом его варианте, ориентированном на дальнейшее использование ПМК, в команду вызова вспомогательного алгоритма внесены некоторые изменения. В скобках после имени вспомогательного алгоритма дается список его параметров (переменных), подразделенных на три группы: аргументы, начинающиеся со слова арг; результаты со слова рез; рабочие регистры rg, используемые во вспомогательном алгоритме для хранения промежуточных данных. Все перечисленные переменные, используемые во вспомогательном алгоритме, перечисляются и в заголовке основного алгоритма после слова нач.

Возможность использования вспомогательного алгоритма в тексте основного будет зависеть от того, как распределены регистры адресуемой памяти ПМК. Если для выполнения основного и вспомогательного алгоритмов для разных переменных будут выделены одни и те же регистры памяти, то решение задачи может стать невозможным. Однако при наличии в ПМК всего 14—15 регистров выделить разные регистры для основного и вспомогательного алгоритмов не всегда возможно. Поэтому приходится ограничиться требованием, чтобы промежуточные данные основного алгоритма, хранящиеся в регистрах, используемых во вспомогательном алгоритме, не пересекались с регистрами, используемыми в основном алгоритме.

тельном, были уже не нужны для работы основного алгоритма в момент обращения к вспомогательному. Поэтому в самой команде обращения к вспомогательному алгоритму необходимо перечислять все регистры адресуемой памяти, используемые при выполнении последнего.

Кроме регистров, используемых для хранения аргументов и результатов вспомогательного алгоритма, в каждой команде вызова должны быть указаны регистры, используемые во вспомогательном алгоритме в качестве рабочих для хранения промежуточных величин. Для упрощения записи команды вызова в последней перечисляются лишь имена (адреса) регистров без указания имен промежуточных величин, для хранения которых они предназначены. При записи команды вызова важно проверить, что использование регистров во вспомогательном алгоритме не оказывает влияния на содержимое тех же регистров при выполнении основного алгоритма.

Перед началом выполнения вспомогательного алгоритма должны быть определены значения всех его аргументов. Последнее реализуется путем использования БЕЙСИК-команд присваивания, помещаемых до или после команды вызова, но до соответствующих команд обращения к подпрограмме ПМК.

7.2. ПОДПРОГРАММЫ

7.2.1. ОСОБЕННОСТИ ДОКУМЕНТАЦИИ НА ПОДПРОГРАММУ

Процесс разработки вспомогательного алгоритма не отличается от разработки основного, а именно:

разработка исходного вспомогательного алгоритма на алгоритмическом языке (лексиконе), ориентированном на пользователя;

детализация вспомогательного алгоритма на промежуточном языке БЕЙСИК-ПМК;

детализация БЕЙСИК-алгоритма (составление программы решения задачи на ПМК);

отладка составленной программы;

разработка программной документации.

Программу, реализующую выполнение вспомогательного алгоритма на ПМК, будем называть *подпрограммой*. Она обычно разрабатывается для многократного использования в основных программах и, как правило, выполняет типовые, стандартные процедуры, не предусмотренные системой команд ПМК. Это процедуры, предписания на выполнение которых оформляются в виде подпрограмм.

Указанные особенности подпрограмм заставляют несколько изменить требования к сопровождающей документации (по сравнению с требованиями к документации на основную программу, изложенными в п. 5.5):

1. На каждую подпрограмму составляется отдельная документация, не включаемая в документацию на основную программу.

2. Для отличия документации на подпрограмму ее начало обозначается словом ПОДПРОГРАММА, а не ПРОГРАММА, как в случае основной программы.

3. Из состава документации на подпрограмму исключается раздел ИНСТРУКЦИЯ. Этот раздел для подпрограммы оказывается ненужным, поскольку пользователь ПМК ее работой не управляет: она запускается и оканчивается автоматически, а все сведения, необходимые для включения подпрограммы в основную программу, приводятся в алгоритме подпрограммы. С учетом этого документация на подпрограмму включает в себя три раздела:

ПОДПРОГРАММА
АЛГОРИТМ (записывается алг)
ТЕСТ

Раздел ПОДПРОГРАММА является ее заголовком, состоящим из номера и имени подпрограммы, устанавливаемых пользователем (автором). В этом же разделе приводятся полное название подпрограммы, краткая постановка задачи, расчетные формулы, а также ограничения, налагаемые на использование подпрограммы. Например:

ПОДПРОГРАММА 7.0 — CNM. Расчет числа сочетаний из n элементов по m :

$$C_n^m = \prod_{i=m}^n \frac{n-m+i}{i}.$$

где $n \geq 1$, $1 \leq m < n$.

Как видим, раздел ПОДПРОГРАММА отличается от раздела ПРОГРАММА одним первым словом.

Раздел АЛГОРИТМ, в котором записываются все три уровня представления вспомогательного алгоритма, по сравнению с записью основного имеет следующие особенности:

1) вместо команды останова вспомогательный алгоритм всегда заканчивается командами: на первом уровне — ВОЗВРАТ; на втором уровне — RETURN; на третьем уровне — В/О. Команды RETURN и В/О помещаются в том же месте алгоритма, где в основном алгоритме подпрограммы ставились команды STOP и С/П;

2) адреса команд подпрограммы записываются во вспомогательном алгоритме всегда с адреса 00, хотя в реальных условиях подпрограммы с указанного адреса почти никогда не размещаются. Такие фиктивные адреса команд подпрограммы (начинающиеся с адреса 00) называются символическими. При размещении команд подпрограммы в памяти ПМК вместе с командами основной программы первые переадресуются (транспонируются) путем добавления к каждому символическому адресу подпрограммы (включая и адреса ссылок, содержащихся в самих командах) адреса первой команды (фактического ее размещения в памяти ПМК).

Раздел ТЕСТ, как и в случае основной программы, предназначается для проверки правильности ввода подпрограммы в память ПМК и выполнения контрольного решения по исходным данным. В отличие от основной программы контрольное решение подпрограммы производится с временной заменой команды В/О на команду С/П, что обеспечивает останов вычислений после выполнения последней команды подпрограммы и считывание результатов контрольного решения.

Рассмотрим примеры разработки вспомогательных алгоритмов и подпрограмм с оформлением программной документации.

7.2.2. ПОДПРОГРАММА ЦЕЛ — ВЫДЕЛЕНИЕ ЦЕЛОЙ ЧАСТИ ЧИСЛА

В системе команд МК-52 и МК-61 имеется команда выделения целой части числа из десятичной дроби — это а. К [x], которая предписывает из содержимого регистра X выделить целую часть. Этой команде соответствует БЕЙСИК-команда RX = E (выражение), предписывающая выделить целую часть числа от значения выражения, стоящего в скобках, и результат поместить в регистр X. В системе команд МК-54 и соответственно Б3-34 и МК-56, к сожалению, отсутствует команда выделения целой части числа. Между тем в научно-технических задачах часто требуется выполнение такой операции. Поэтому для МК-54 необходима соответствующая подпрограмма. Для простоты ограничим свою задачу выделением целой части только из положительных чисел.

Постановка задачи. Дано: N — положительная правильная или неправильная десятичная дробь.

Требуется: Выделить ее целую часть E (N).

Решение. Выполнение подпрограммы основано на использовании свойства команд с косвенной адресацией: при обращении к ней отбрасывается дробная часть содержимого индексного регистра. Это означает, что, для того чтобы выделить целую часть числа N, его следует поместить в индексный регистр Ri, используемый при выполнении команды с косвенной адресацией. Если это будет команда чтения, то в результате ее выполнения в регистр X будет помещено некоторое число M (E (N)), значение которого является содержимым того регистра памяти, номер которого совпадает с целой частью N. Нас же будет интересовать не число M, а содержимое индексного регистра Ri. Прочитав его содержимое после выполнения команды с косвенной адресацией, получим искомое значение целой части числа N. Сказанное справедливо для команд с косвенной адресацией, в которых адрес не модифицируется, т. е. при использовании в качестве индексных регистров R7, R8, R9, RA, RB, RC, RD, RE.

Следует отметить, что команда ПМК с косвенной адресацией устойчиво отбрасывает дробную часть содержимого индексного регистра лишь в том случае, когда целая часть больше или равна единице. Если она меньше единицы, то такое отбрасывание не

всегда происходит. Поэтому перед обращением к команде с косвенной адресацией в индексный регистр следует добавить единицу, а после обращения — вычесть.

С учетом изложенного документация на подпрограмму ЦЕЛ будет выглядеть следующим образом:

ПОДПРОГРАММА 7.1 — ЦЕЛ. Выделение целой части числа $N_{\text{д}} = E(N)$, где N — исходное число. алг ЦЕЛ арг RX = N рез RX = E (N) нач R7 = N, R7 = N _д N = N + 1 знач = M [E (N)] N _д = E (N) — 1 возврат кон TEST 1) ВВ RX = 635.678; НЖ КНОП В/О С/П; ВЫ 635 (3 с) 2) ВВ Rx = 0.333; НЖ КНОП В/О С/П; ВЫ 0 (3 с) 3) ВВ RX = 1; НЖ КНОП В/О С/П; ВЫ 1 (3 с)	1 R7 = RX + 1 2 RX = RX [R7] 3 RX = R7 — 1 4 RETURN 00. 1 (1, N) 01. + (N + 1) 02. xP7 (R7 = N + 1) 03. КпX7 04. пX7 (R7) 05. 1 (1, R7) 06. — (R7 — 1) 07. В/О
---	---

7.2.3. ПОДПРОГРАММА СНМ — ОПРЕДЕЛЕНИЕ ЧИСЛА СОЧЕТАНИЙ ИЗ n ЭЛЕМЕНТОВ ПО m

Постановка задачи. Дано: Множество из n элементов.

Требуется: Определить число сочетаний из n элементов по m .

Решение. Как известно, число сочетаний из n элементов по m

$$C_n^m = \frac{A_n^m}{P_m} = \frac{n(n-1)\dots(n-m+1)}{1 \cdot 2 \dots (m-1)m},$$

или

$$C_n^m = \frac{n(n-1)\dots(n-m+1)}{m(m-1)\dots2 \cdot 1} = \prod_{i=m}^1 \frac{n-m+i}{i}, \quad (7.1)$$

где A — число размещений из n элементов по m ; P — число перестановок из m элементов.

Вычисление числа сочетаний на ПМК по рекуррентной формуле (7.1) более удобно, чем по формуле с факториалами:

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

Формула (7.1) значительно уменьшает количество выполняемых операций, поэтому вычисления в подпрограмме *CNM* будут выполняться именно по этой формуле.

По определению $0! = 1$. Это означает, что $C_n^0 = 1$. Легко видеть, что при вычислении C_n^0 по формуле (7.1) будет иметь место некорректная операция — деление на нуль. В этом недостаток формулы (7.1). Поскольку этот недостаток в большинстве случаев несуществен, гораздо проще наложить ограничения на использование подпрограммы при $m = 0$.

С учетом изложенного документацию на подпрограмму можно записать так:

ПОДПРОГРАММА 7.2 — CNM. Расчет числа сочетаний из n элементов

$$\text{по } m \ C_n^m = \prod_{i=m}^1 (n - m + i)/i, \text{ где } n \geq 1, 1 \leq m < n.$$

алг *CNM*

арг R1 = n , R2 = m
рез RY = C_n^m

нач RX = C
 $C = 1$

1 RX = 1

00. 1 (1)

нц $C = C * n/m$

2 RX = RX * R1/R2

- 01. пX1 (R1, C)
- 02. пX2 (R2, R1, C)
- 03. \div (R1/R2, C)
- 04. \times ($C * R1/R2$)

$n = n - 1$

3 UNTIL R1 = 1 GOTO 4

- 05. FL1
- 06. 07

кц
до $m = 1$ от m шаг -1

4 UNTIL R2 = 1 GOTO 2

- 07. FL2
- 08. 01

ВОЗВРАТ

5 RETURN

09. B/O

кон
ТЕСТ
BB R1 = $n = 6$, R2 = $m = 2$
НЖ В/О С/П; ВЫ $C_6^2 = 15$ (12 с)

В алгоритме используются две БЕЙСИК-команды условного перехода по счетчику. Первая из них 3 UNTIL R1=1 GOTO 4 для организации повторений не используется. После ее выполнения из содержимого регистра R1 вычитается единица и во всех случаях осуществляется переход к следующей БЕЙСИК-команде, т. е. эта команда уменьшает значение n с каждым циклом на единицу и играет роль простого счетчика. Вторая команда 4 UNTIL R2=1 GOTO 2 помимо уменьшения значения на единицу организует повторения путем передачи управления в каждом из них БЕЙСИК-команде 2. Как только значение m станет равным единице, происходит выход из повторений и возврат к выполнению основной программы. Соответственно выполняются и команды подпрограммы 05. FL1 06. 07 и 07.FL2 08. 01.

7.2.4. ПОДПРОГРАММА БРВ — БИНОМИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ ВЕРОЯТНОСТЕЙ

Постановка задачи. Дано: Произведено n опытов. Вероятность появления некоторого события A в каждом из них равна p .

Требуется: Определить вероятность появления события A ровно m раз из серии в n опыта.

Решение. Как известно, расчет искомой вероятности производится по формуле Бернулли

$$P_n^m = C_n^m p^m (1 - p)^{n-m}, \quad (7.2)$$

где P_n^m — вероятность того, что событие A появится ровно m раз в серии из n опытов.

При выполнении вычислений по формуле (7.2) можно было бы воспользоваться подпрограммой *CNM*, определяющей C_n^m . Однако если несколько изменить команды указанной подпрограммы, возложив на нее вычисление не только C_n^m , но и произведения $C_n^m * p^m$, то можно существенно сократить длину подпрограммы БРВ. Для этого данное произведение необходимо представить следующим образом:

$$C_n^m p^m = \prod_{i=m}^1 \frac{n - m + i}{i} = \prod_{i=m}^1 \frac{n - m + i}{i} p.$$

Команда повторения, вычисляющая значение произведения $C_n^m p^m$, будет иметь вид

```
C = 1
нц
|   C = (C * n/m) * p
|   n = n - 1
кц
до m = 1 от m шаг -1.
```

Для реализации этой команды в тело цикла достаточно будет ввести дополнительно две команды, которые записываются в последовательности команд подпрограммы *CNM* после команды с адресом 04:

- 05. пX6 (p , $C * n/m$)
- 07. \times ($p * C * n/m$)

При составлении подпрограммы БРВ следует сначала предусмотреть вычисление выражения $(1 - p)^{n-m}$ по команде возвведения в степень, а затем определить $C_n^m p^m$. Это необходимо потому, что после вычисления $(1 - p)^{n-m}$ по команде возвведения в степень значения n и m сохраняются, а после вычисления C_n^m теряются. Для вычисления C воспользуемся командами ранее разработанной подпрограммы *CNM* с необходимыми изменениями для получения произведений $C_n^m p^m$.

С учетом изложенного документацию на подпрограмму можно представить в следующем виде:

ПОДПРОГРАММА 7.3—БРВ. Биномиальное распределение вероятности

$$P = C_n^m p^m (1 - p)^{n-m},$$

где p — вероятность того, что событие появится при одном испытании ($0 \leq p \leq 1$); n — число испытаний ($n \geq 1$); m — число появлений ожидаемого события, равное m в серии из n испытаний ($1 \leq m \leq n$); P_n^m — вероятность того, что в серии из n испытаний ожидаемое событие появится ровно m раз.

алг БРВ
 арг R1 = n , R2 = m , R6 = p
 рез RX = P_n^m
 нач RY = C , R7 — степень
 $\text{степень} = (1 - p)^{n-m}$

```

    1 R7 = (1 - R6) * * (R1 - R2)
        00. пX1 (R1)
        01. пX2 (R2, R1)
        02. - (R1 - R2)
        03. 1 (1, R1 - R2)
        04. пX6 (R6, 1, R1 - R2)
        05. - (1 - R6, R1 - R2)
        06. FXb ((1 - R6) * * (R1 - R2))
        07. xП7 (R7 = степень)
    
```

$C = 1$
 2 RX = 1
 08. 1 (1)

нц $C = C (* n/m) * p$
 3 RX = RX * R6 * R1/R2

```

        09. пX1 (R1, C)
        10. пX2 (R2, R1, C)
        11. - (R1/R2, C)
        12. пX6 (R6, R1/R2, C)
        13. X (R6 * R1/R2, C)
        14. X (C * R6 * R1/R2)
    
```

$n = n - 1$
 4 UNTIL R1 = 1 GOTO 5

```

        15. FL1
        16. 17
    
```

кц
 до $m = 1$ от m шаг —1
 5 UNTIL R1 = 1 GOTO 3

```

        17. FL2
        18. 09
    
```

$P_n^m = (C_n^m * p^m) * \text{степень}$
 6 RX = RX * R7

```

        19. пX7
        20. X
    
```

ВОЗВРАТ
 7 RETURN

кон
 ТЕСТ
 ВВ R1 = $n = 4$, R2 = $m = 2$, R6 = 0.25
 НЖ В/О С/П; ВЫ $P_n^m = 2.1093749 -01$ (12 с)

7.2.5. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ В ТЕКСТЕ ОСНОВНОЙ ПРОГРАММЫ

Запись команд подпрограммы в тексте основной программы.
 В процессе выполнения команд основной программы должны быть
 также выполнены и все команды входящих в нее подпрограмм.
 Команды всех подпрограмм, необходимых для выполнения основ-

ной программы, должны быть введены в память ПМК одновре-
 менно с вводом команд основной программы. Это требует записи
 команд подпрограмм в тексте основной программы.

Поскольку на каждую подпрограмму разрабатывается отдель-
 ная документация, нет необходимости приводить ее полностью
 в документации на основную программу. Достаточно представить
 ее в таком виде, чтобы она в нужном месте была введена в память
 ПМК. Поэтому последовательность команд каждой подпрограммы
 записывается в документации на основную программу только
 в компактной форме, как это показано в п. 5.5 (программа 5.2 К).
 Исходный и БЕЙСИК-алгоритмы при этом не приводятся.

Подпрограмма рассматривается как БЕЙСИК-команда, реали-
 зуемая всей последовательностью команд подпрограммы, записан-
 ной на входном языке программирования ПМК. Такая БЕЙ-
 СИК-команда называется SUB (от английского слова *subroutine* —
 подпрограмма) и записывается в следующем виде:

номер SUB [имя подпрограммы]
 команды подпрограммы в компактной форме

31. 1	33. xП7	35. пX7	37. —
32. +	34. КпX7	36. 1	38. В/О

12 SUB

Номер подпрограммы является очередным возрастающим но-
 мером БЕЙСИК-команды, а сама команда SUB — заголовком под-
 программы, записываемой под ним в виде последовательности
 команд ПМК. Номер записывается в той же позиции, что и номера
 всех остальных БЕЙСИК-команд алгоритма. Адреса команд ПМК
 подпрограммы должны быть при записи транспонированы (пере-
 адресованы) из символьических в истинные, т. е. увеличены все
 на значение адреса, соответствующего расположению нулевой
 команды подпрограммы в основной программе. В приведенном
 примере подпрограмма 12 SUB располагается в основной про-
 грамме с адреса 31.

Внутренняя и внешняя подпрограммы. Различают два способа
 включения подпрограмм в основную программу: внутри текста
 основной программы и вне его, после команды С/П основной про-
 граммы. Подпрограмму, включенную в основную программу
 по первому способу, условимся называть внутренней, а включен-
 ную по второму способу, — внешней. Текст внутренней подпро-
 граммы в нужном месте основной программы помещается полно-
 стью, за исключением команды В/О. При этом, как всегда, сим-
 вольеские адреса подпрограммы заменяются на истинные в соот-
 ветствии с местом расположения подпрограммы. Использование
 подпрограммы в основной программе в качестве внутренней
 уменьшает объем занимаемой программной памяти по крайней
 мере на 3 ячейки (исключаются команды обращения к подпро-

рамме и команда В/О, всего три шага). Однако такой выигрыш возможен лишь тогда, когда в основной программе записано только одно обращение к данной подпрограмме. В случае необходимости многократного обращения пришлось бы многократно записывать текст подпрограммы в текст основной программы, что неоправданно увеличило бы длину последней. Поэтому при многократном использовании подпрограммы ее целесообразно оформить как внешнюю.

Текст внешней подпрограммы может быть записан в любом свободном участке памяти ПМК, но, как правило, записывается после окончания основной программы. Последней командой внешней подпрограммы должна быть команда В/О, предписывающая продолжение выполнения основной программы.

Команда обращения к внешней подпрограмме. В системе команд ПМК имеется двухшаговая команда обращения к подпрограмме $a + 0$. ПП $a + 1. nn$, где ПП означает подпрограмму, а nn — адрес размещения нулевой команды подпрограммы в памяти ПМК. Указанной команде соответствует БЕЙСИК-команда обращения к подпрограмме: номер GOSUB номер команды-подпрограммы. Например, 4GOSUB 12. Слово GOSUB происходит от двух английских слов: *go* — идти и *subroutine* — подпрограмма. Внутри подпрограммы допускаются также обращения и к другим внешним подпрограммам. Глубина вложенности обращений — до пяти включительно.

Шаблон команды вызова вспомогательного алгоритма,

1. Вызов внутреннего алгоритма и подпрограммы:

имя алгоритма (арг регистр 1 = имя [, регистр 2 = имя ...],
рез регистр 3 = имя [, регистр 4 = имя ...],
[пр список рабочих регистров])
 $A - B$ — команды присваивания значений аргументам
 B — имя подпрограммы
 $a + 1$. Команды подпрограммы в компактной форме
 $B + 1$ следующая B -команда основного алгоритма
 $c + 0$ следующая команда P -программы.

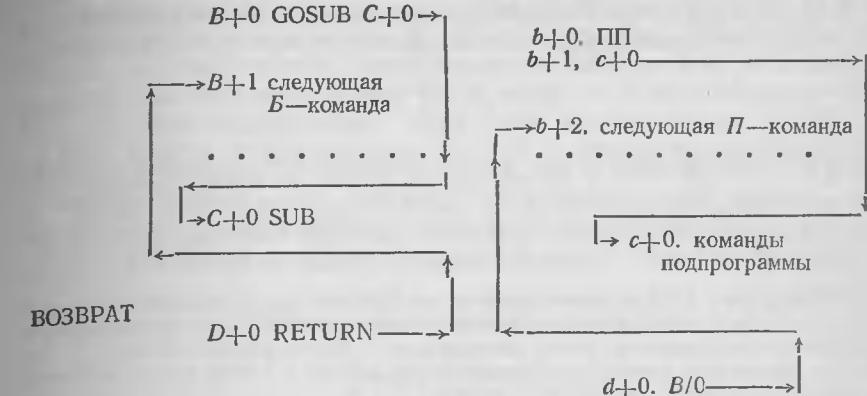
Пример:
ЦЕЛ (арг RX = N, рез RX = E (N))
12 SUB ЦЕЛ

31. 1 33. xP7 35. pX7 37. —
32. + 34. KpX7 36. 1

13 следующая B -команда основного алгоритма
38. следующая P -команда основной программы.

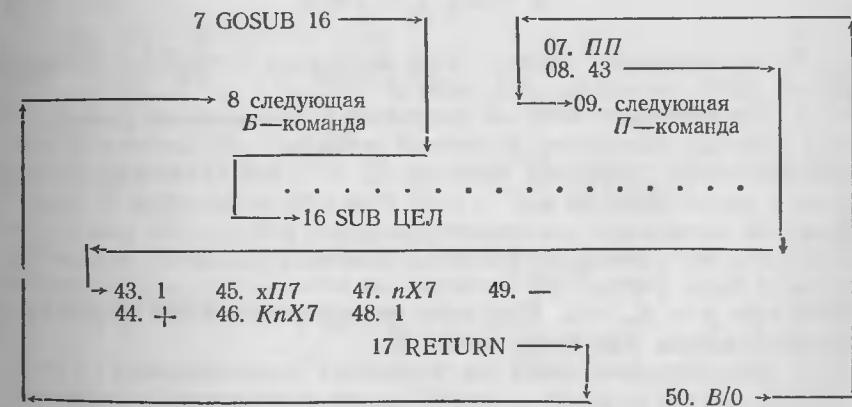
2. Вызов внешнего алгоритма:

имя алгоритма (арг регистр 1 = имя [, регистр 2 = имя...],
рез регистр 3 = имя [, регистр 4 = имя...],
[пр список регистров])
 $A + 0$ команды присваивания значений аргументам $a + 0$.
впиши аргументов в память



Пример:

ЦЕЛ (арг RX = N, рез RX = E (N), пр R7)



7.2.6. ПРИМЕР ИСПОЛЬЗОВАНИЯ ПОДПРОГРАММ В ТЕКСТЕ ОСНОВНОЙ ПРОГРАММЫ. ПРОГРАММА РАСЧЕТА МАТЕМАТИЧЕСКОГО ОЖИДАНИЯ ЧИСЛА ИСПРАВНЫХ ТЕЛЕФОННЫХ КАНАЛОВ

Постановка задачи. Дано: Направление связи, состоящее из n одинаковых телефонных каналов. Известна характеристика надежности каждого канала, коэффициент готовности канала K_i (вероятность того, что в любой момент времени выделенный канал будет исправен и пригоден для использования).

Требуется: Определить математическое ожидание числа готовых к использованию телефонных каналов K по формуле

$$K = E \left(\sum_{m=0}^n P_n^m m \right), \quad (7.3)$$

где m — число пригодных к использованию телефонных каналов; n — общее число имеющихся каналов; P_n^m — вероятность того, что из n каналов будет пригодно к использованию ровно m каналов; E — целая часть числа, заключенного в скобках.

Решение. Как это следует из постановки задачи, расчет значения K производится с использованием¹ формулы Бернулли (7.2). Для этой цели желательно воспользоваться ранее разработанной подпрограммой БРВ—7.3. Поэтому необходимо рассмотреть, можно ли это сделать с учетом тех ограничений, которые налагаются на исходные значения P и m .

1. Подпрограмма БРВ не допускает использования значений $m = 0$. Если $m = 0$, то $P_n^m * m = 0$. Учитывая это, в основной программе при расчете по формуле (7.3) суммирование следует начать не с нуля, а с единицы и формулу (7.3) преобразовать к виду

$$K = E \left(\sum_{m=1}^n P_n^m m \right). \quad (7.4)$$

Тогда значения P можно будет вычислять с помощью подпрограмм БРВ, положив при этом $p = K_r$.

2. Подпрограмма БРВ не допускает использования $p = K_r = 1$ (случай абсолютно надежных каналов). Но очевидно, что при абсолютно надежных каналах $K_r = 1$, все каналы будут готовы к использованию и $K = n$. В этом случае значение K можно будет не вычислять по подпрограмме, а в основной программе положить его равным n . Иными словами, в основной программе должно быть учтено ограничение, налагаемое на подпрограмму БРВ при $p = K_r = 1$. При этом подпрограмма БРВ не должна использоваться для вычисления K_r .

3. Подпрограмма БРВ не допускает использования $p = 0$. Но этот случай не имеет в данной задаче практического значения, так как при $p = K_r = 0$ (абсолютно ненадежные каналы) число готовых к использованию каналов заведомо равно нулю.

С учетом ограничений, налагаемых подпрограммой БРВ, и требований, налагаемых постановкой задачи, эта подпрограмма может быть использована для расчета математического ожидания числа пригодных каналов связи. Поскольку она многократно должна быть использована в основной программе, ее целесообразно оформить как внешнюю подпрограмму.

Исходные формулы расчета математического ожидания (7.3) и (7.4) требуют выполнения операции выделения целой части числа (число каналов должно быть всегда только целым). В отличие от вычислений по формуле Бернулли (определение P_n^m) операция

выделения целой части числа должна быть выполнена в самом конце основной программы и только один раз. Для этой цели можно воспользоваться подпрограммой ЦЕЛ—6.1, предназначенной для выделения целой части числа, и использовать ее как внутреннюю подпрограмму. В случае использования МК-52 и МК-61 вместо подпрограммы ЦЕЛ-6.1 можно воспользоваться командой выделения целой части числа $a + 0$. $K [x]$.

В соответствии со сделанными замечаниями документация на программу расчета ожидаемого числа пригодных к использованию каналов связи будет иметь следующий вид:

ПОДПРОГРАММА 7.4 — МОК. Расчет математического ожидания числа пригодных к использованию каналов связи $K = f(n, K_r)$

$$K = E \left(\sum_{m=1}^n P_n^m m \right),$$

где n — общее число имеющихся в наличии каналов ($n \geq 1$); m — число пригодных к использованию каналов ($1 \leq m \leq n$); P_n^m — вероятность того, что из n каналов будут пригодны к использованию ровно m каналов,

$$P_n^m = C_n^m K_r (1 - K_r)^{n-m},$$

где K_r — коэффициент готовности канала, $0 < K_r \leq 1$.

```
алг MOK
    арг R4 = n, R6 = K_r
    рез RX = K
нач R3 = m, R8 = S, пр R1, R2, R7
    BB R4 = n, R6 = K_r
    если (1 - p) ≠ 0
        1 RX = 1 - R6
        00. 1 (1)
        01. пX6 (R6, 1)
        02. -(1-R6)
        2 IF RX ≠ 0 ELSE 14
            03. Fx ≠ 0
            04. 32
        to S = 0
            3 R8 = 0
            05. 0 (0)
            06. xΠ8 (R8 = 0)
        m = n
            4 R3 = R4
            07. пX4 (R4)
            08. xΠ3 (R3 = R4)
    иц
        БРВ (апр R1 = n, R2 = m, R6 = K_r, рез Rx = P_n^m пр R7)
        5 R2 = R3
        09. пX3 (R3)
        10. xΠ2 (R2 = R3)
        6 R1 = R4
        11. пX4 (R4)
        12. xΠ1 (R1 = R4)
        7 GOSUB 16
        13. ПП
        14. 34
```

знач = $m * P_n^m$
 8 R3 = R3 * RX
 S = S + знач
 9 R8 = R8 + RX

кц
до $m = 1$ от m шаг —1

10 UNTIL R3 = 1 GOTO 5
 20. FL3
 21. 09

знач = S

11 RX = R8
 22. пX8 (S)

ЦЕЛ (арг RX = N, рез RX = E (N), пр R7)

12 SUB ЦЕЛ
 23. 1 25. xP7 27. пX7 29. —
 24. + 26. KпX7 28. 1
 иначе $K = n$

13 GOTO 15
 30. БП
 31. 33

14 RX = R4
 32. пX4 (R = n)

15 STOP
 33. С/П (K)

16 SUB БРВ
 34. пX1 38. цX6 42. 1 46. пX6 50. 51 54. X
 35. пX2 39. — 43. пX1 47. X 51. FL2 55. В/О
 36. — 40. FX^y 44. пX2 48. X 52. 43
 37. 1 41. xP7 45. ÷ 49. FL1 53. пX7

ИНСТРУКЦИЯ

ВВ R4 = n, R6 = K_р

ВЫ K

ТЕСТ

- 1) ВВ R4 = n = 6, R6 = K = 0,5
 НЖ В/О С/П; ВЫ K = 2 (2,5 мин)
- 2) ВВ R4 = n = 4, R6 = K = 1
 НЖ В/О С/П; ВЫ K = 4 (2 с)

В приведенной программе МОК использованы две подпрограммы: одна из них ЦЕЛ используется как внутренняя, а другая БРВ — как внешняя. Последняя помещена за пределами основной программы, после ее команды останова 33.С/П. В конце подпрограммы БРВ помещена команда возврата к продолжению выполнения основной программы 55.В/О. Эта команда предписывает перейти к выполнению команды 15.пX3 основной программы, которая непосредственно следует за командой вызова подпрограммы 13.ПП 14.34, где 34 является адресом начальной команды подпрограммы БРВ.

Перед обращением к подпрограмме (БЕЙСИК-команда 7 GOSUB 16) производится присваивание значений ее аргументам R2 = R3 = m и R1 = R4 = n. Присваивания значения R6 = p = K_р не требуется, так как оно уже было присвоено при вводе значения K_р перед запуском программы. Обращение к внешней подпрограмме БРВ повторяется в основной программе m раз. В каждом цикле повторения перед обращением к подпрограмме (7 GOSUB 16) значения аргументам подпрограммы (БЕЙСИК-команды 5 и 6) присваиваются вновь. Вызвано это тем, что значения аргументов после выполнения подпрограммы, как правило, теряются, и их перед каждым новым обращением необходимо либо восстановить (например, n, 6R1 = R2), либо присвоить им новые значения (m, 5 R2 = R3).

Внутренняя подпрограмма ЦЕЛ записана в основном алгоритме с помощью БЕЙСИК-команды 12 SUB. В ней отсутствует команда В/О. Последней ее командой является 29. —, после которой выполняется очередная команда основной программы: 30. БП 31. 33, предписывающая выполнение безусловного перехода после окончания ветви то.

Проверка правильности использования регистров памяти ПМК при составлении программ, включающих подпрограммы. После окончания записи алгоритма и программы необходимо произвести проверку использования регистров памяти ПМК. При этом проверке подлежат все регистры памяти ПМК, выделенные для хранения аргументов, результатов и промежуточных величин как в основной программе, так и во входящих в нее подпрограммах. Проверка осуществляется в следующем порядке.

1. Определяется общий список адресуемых регистров, используемых при выполнении программы. Для этого в список основного алгоритма после служебного слова нач дополнительно записываются регистры, используемые в подпрограммах. Допускается при этом не указывать имена переменных подпрограмм, а перечислить в дополнительном списке только имена регистров, снабдив начало дополнительного списка служебным словом прг. Такими регистрами в нашем примере являются регистры R1 (для хранения значений n в подпрограмме БРВ), R2 (для хранения значения m) и R7 (для хранения промежуточных величин в подпрограммах БРВ и ЦЕЛ). В результате в заголовке основного алгоритма будет помещен полный список регистров, используемых при выполнении основной программы, включая и подпрограммы, к которым предусмотрено обращение.

2. Определяются адресуемые регистры, используемые для хранения значений более чем одной переменной. Для этого проверяется каждый адресуемый регистр, записанный в заголовке основного алгоритма после служебных слов арг, рез, нач, независимо от того, где и как он используется в основной программе или подпрограммах. Таким регистром в рассмотренной программе МОК является регистр R7. Первый раз он используется в под-

программе БРВ для хранения промежуточных результатов (степень и $C_n^m p$), а второй раз — в подпрограмме ЦЕЛ для выделения целой части числа.

3. Устанавливается отсутствие нежелательных последствий использования адресуемых регистров для хранения двух и более переменных. В рассмотренном случае после выполнения подпрограммы БРВ содержимое регистра R7 нигде и никак не используется. Поэтому оно может быть стерто или вместо него может быть записано значение другой переменной. В конце основной программы в регистр R7 помещается значение суммы S , из которой подпрограммой ЦЕЛ выделяется целая часть, равная исковому числу каналов K . Такое использование регистра R7 в программе вполне допустимо и не вызывает никаких нежелательных последствий.

Упражнения

7.2.1. Составить подпрограмму вычисления $n!$ с учетом необходимости выдачи значения $0! = 1$.

7.2.2. Составить подпрограмму вычисления $C_n^m = n!/(m!(n-m)!)$, используя подпрограмму вычисления $n!$, составленную при выполнении упражнения 7.2.1. Какими преимуществами будет обладать эта подпрограмма по сравнению с подпрограммой CNM — 7.2, рассмотренной в п. 7.2.3?

7.2.3. Разработать новую подпрограмму БРВ, устраняющую основные недостатки подпрограммы БРВ — 7.3.

7.3. РАЗРАБОТКА ПРОГРАММНЫХ КОМПЛЕКСОВ ИЗ РАНЕЕ СОСТАВЛЕННЫХ ПОДПРОГРАММ И ПРОГРАММ

7.3.1. ПОНЯТИЕ О ПРОГРАММНЫХ КОМПЛЕКСАХ

Вновь разрабатываемых программах могут быть использованы не только ранее составленные подпрограммы, но и целые программы и/или их составные части (блоки). Программы-компоненты или их составные части, включаемые во вновь разрабатываемую программу из программного задела, условимся для краткости называть модулями. В качестве модулей могут выступать также и подпрограммы.

Программным комплексом будем называть вновь разрабатываемую программу, включающую в себя не менее двух модулей. Очевидно, что помимо модулей программный комплекс должен включать в себя кросспрограмму, осуществляющую объединение модулей в единый комплекс и организующую взаимодействие между ними. Поэтому разработка программного комплекса будет включать:

компоновку программного комплекса из готовых модулей; разработку кросспрограммы.

Обращение к модулям в алгоритме программного комплекса, а также запись программ модулей в последовательности команд комплексной программы производится при этом так же, как и запись подпрограмм в основной программе. При этом каждому модулю должно быть присвоено свое имя. Желательно после имени модуля через дефис указывать ссылку на источник, откуда взят этот модуль. Обычно это номер, под которым зарегистрирован алгоритм или программа (подпрограмма) в личной библиотеке пользователя.

7.3.2. ПОРЯДОК РАЗРАБОТКИ ПРОГРАММНЫХ КОМПЛЕКСОВ

Разработка программных комплексов включает в себя следующие этапы:

1. Постановка задачи.
2. Определение состава модулей, включаемых в программный комплекс.
3. Разработка алгоритма программного комплекса, включающая:

составление кросспрограммы, объединяющей используемые модули в программный комплекс;

оценку возможности размещения программного комплекса в памяти ПМК;

проверку корректности использования регистров адресуемой памяти ПМК в программном комплексе.

4. Запись полного текста программы комплекса в виде сводной последовательности выполняемых команд. Отладка программы комплекса.

5. Разработка документации на программный комплекс.

Рассмотрим разработку программного комплекса на примере расчета вероятности обслуживания заявок на телефонные переговоры в случае не абсолютно надежных каналов связи.

7.3.3. ПОСТАНОВКА ЗАДАЧИ

1. Формулировка задачи. Да и о: 1. Направление связи, состоящее из n равнодоступных и однородных телефонных каналов.

2. Надежность каждого канала, определяемая коэффициентом его готовности K (вероятности того, что к моменту поступления заявки на переговоры канал окажется готовым к работе).

3. Простейший поток заявок на переговоры, поступающий на узел связи, с приведенной плотностью $\alpha = \lambda/\mu$, где λ — интенсивность поступления заявок, измеряемая числом заявок в единицу времени, $\lambda = 1/\bar{t}_3$, где \bar{t}_3 — среднее время между моментами поступления заявок; μ — интенсивность обслуживания заявок; $\mu = 1/\bar{t}_n$, где \bar{t}_n — средняя продолжительность телефонных переговоров.

Требуется: Разработать алгоритм вычисления на ПМК вероятности обслуживания заявок на телефонные переговоры:

$$P_{\text{обс}} = f(n, K_r, \alpha).$$

Результат расчета оформить в виде таблицы:

n	K_r	α	K	$P_{\text{обс}}$
Исходные данные			Результаты	

Указание. Использовать ранее разработанные программы для вычислений: $K = f_1(n, K_r)$ (программа МОК — 7.4); $P_{\text{обс}} = f_2(K, \alpha)$ (программа ЭРЛ — 6.1).

Определение состава модулей, включаемых в программный комплекс. Решение сформулированной задачи осуществляется по двум ранее разработанным программам (которые целесообразно без изменений включить в состав программного комплекса в качестве его модулей):

расчета числа каналов, готовых к использованию $K = f_1(n, K_r)$ (программа МОК — 7.4; длина ее — 56 шагов);

расчета вероятности обслуживания заявок на телефонные переговоры для абсолютно надежных каналов по формуле Эрланга $P_{\text{обс}} = f_2(K, \alpha)$ (программа ЭРЛ — 6.1; длина ее — 25 шагов).

Общая длина модулей программного комплекса $56 + 25 = 81$ шаг. Для кросспрограммы с учетом возможности ее размещения в МК-54 останется объем программной памяти, равный $98 - 81 = 17$ ячеек. Оба включаемых модуля — внутренние. В последовательности команд кросспрограммы обращение к каждому из модулей будет записываться только один раз. Поэтому в конце каждого модуля уже не будет необходимости ставить команду С/П (команда С/П будет предусмотрена в кросспрограмме как общая команда останова программного комплекса). Тогда число команд в каждом из модулей может быть уменьшено на единицу.

Однако модуль МОК — 7.4 включает в себя подпрограмму БРВ, помещенную после команды останова основной программы. Если просто убрать из модуля МОК — 7.4 команду останова, то после окончания последовательности команд основной программы будет автоматически выполняться подпрограмма БРВ (без обращения к ней), что недопустимо. Для того чтобы исключить паразитное выполнение подпрограммы БРВ, следует обойти ее с помощью двухшаговой команды БПп (где pp — адрес команды основной программы, помещаемой вслед за последней командой подпрограммы). В результате длина модуля МОК по сравнению с программой МОК — 7.4 увеличится на один шаг и составит 57 шагов. Тогда для размещения кросспрограммы останется $98 - 81 = 17$ ячеек памяти.

7.3.4. РАЗРАБОТКА АЛГОРИТМА ПРОГРАММНОГО КОМПЛЕКСА

Порядок выполнения вычислений в программном комплексе. Установим следующий порядок вычислений в программном комплексе:

1. Ввод исходных данных.
2. Запуск модуля МОК. Исходные данные: n и K_r ($RX = K$).
3. Запуск модуля ЭРЛ. Исходные данные: $RX = K$ и $RA = \alpha$. Результат расчета $RX = P_{\text{обс}}$. (Переход от модуля МОК к модулю ЭРЛ должен выполняться без останова вычислений.)
4. Останов вычислений и выдача результатов решения задачи, выполняемой программным комплексом ($RX = K$ и $RY = P_{\text{обс}}$).

Из изложенного следует, что весь процесс вычислений в рассматриваемом программном комплексе выполняют в основном входящие в него модули МОК — 7.4 и ЭРЛ — 6.1. На кросспрограмму, объединяющую указанные модули в программный комплекс, возлагаются в основном функции управления вычислениями. К таким функциям относятся:

1. Переход от модуля МОК к модулю ЭРЛ без останова.
2. Выдача значения K и $P_{\text{обс}}$ после окончания работы программного комплекса.

3. Разветвление вычислений при $K = 0$ и $K \neq 0$.

Очевидно, что модуль МОК при $n = 1$ и $K_r < 1$ всегда даст результат $K = 0$, поскольку округление математического ожидания числа каналов производится всегда в меньшую сторону. Тогда модуль ЭРЛ, для которого допустимы только значения $K = 1, 2, \dots, n$, не может быть использован для расчета $P_{\text{обс}}$. Поэтому при $K = 0$ необходимо исключить из схемы вычислений модуль ЭРЛ, направив вычисления по другой ветви. Сделаем это следующим образом. При $K = 0$ в модуль ЭРЛ введем в качестве исходного данного $K = n = 1$ и полученное значение вероятности $P_{\text{обс}}$ умножим на K_r . Здесь будет использована теорема умножения вероятностей. Вероятность совместного появления двух событий (события, когда единственный канал готов к обслуживанию заявки, и событие, когда он не занят) равна произведению вероятностей появления этих событий. Можно показать, что все возможные сочетания этих событий (готов — свободен, готов — занят, не готов), составляют полную группу событий, сумма вероятностей которых составляет единицу, и, следовательно, применение теоремы умножения вероятностей правомерно.

Разработка алгоритма программного комплекса и кросспрограммы. Поскольку предполагается, что алгоритмы и программы модулей были разработаны и оформлены надлежащим образом ранее (до составления программного комплекса), то разработка последнего может быть сведена в основном к разработке кросспрограммы и алгоритма. Здесь приобретает особую важность вопрос размещения кросспрограммы и модулей в ограниченной памяти ПМК. Другим важным вопросом при разработке комплекса является корректность использования адресуемых регистров па-

мия в различных частях программируемого комплекса. Многократное использование одних и тех же регистров адресуемой памяти в кросспрограмме и модулях для хранения различных данных не должно приводить к искажению последних в процессе выполнения программного комплекса.

Для решения указанных вопросов предварительно составляется упрощенный алгоритм программного комплекса, в котором записываются команды на всех трех уровнях представления только для кросспрограммы. Модули при этом представляются командами обращения к ним, а П-команды модулей — начальными и конечными их адресами. Тогда упрощенный алгоритм программного комплекса может быть представлен в виде

```

алг обслуживание
    арг R4 = n, R6 = Kr, RA = α
    рез RX = K, RY = Pобс
нач
    R9 = K, pr R1, R2, R3, R5, R7, R8
    BB R4 = n, R6 = Kr, RA = α
    НЖ В/О С/П
    MOK — 7.4 (арг R4 = n, R6 = Kr, рез RX = K, pr R1, R2, R3, R7, R8)
        1 SUB MOK-7.4
        00. } MOK = 7.4
        .. }
        56. }

K = знач
    2 R9 = RX
    57. xP9 (R9 = K)

если K = 0
    3 IF RX = 0 ELSE 5
        58. Fx = 0
        59. 61

    то K = 1
    4 RX = 1
    60. 1

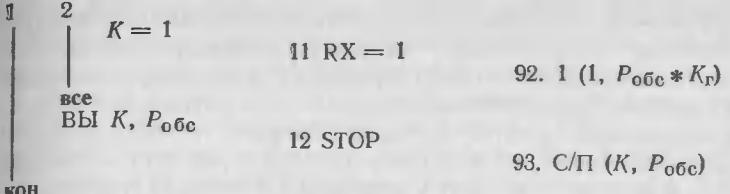
все
ЭРЛ — 6.1 (арг R3 = K, RA = α, рез RX = Pобс, pr R1, R2, R5)
    5 R3 = RX
    61. nX3 (R3)

    6 SUB ЭРЛ-6.1
    62. } ЭРЛ — 6.1
    .. }
    85. }

если K = 0
    7 RX = R9
    86. nX9 (K, Pобс)
    8 IF RX = 0 ELSE 11
        87. Fx = 0
        88. 93

    то Pобс = Pобс * Kr
    9 RX = RY
    89. F. (Pобс)
    10 RX = RX * R6
    90. nX6 (Kr, Pобс)
    91. x (Pобс, Kr)
    11 RX = 1
    все
    ВЫ K, Pобс
кон

```



Оценка возможности размещения программного комплекса в памяти ПМК. В результате разработки упрощенного алгоритма и кросспрограммы устанавливается длина программного комплекса и объем программной памяти, необходимый для его размещения. В рассматриваемом случае длина программного комплекса будет составлять 94 шага, и он может быть размещен в памяти ПМК всех типов. Однако после составления упрощенного алгоритма и кросспрограммы иногда выясняется, что длина программного комплекса превышает допустимые пределы, обусловленные объемом программной памяти ПМК. В этом случае необходимо принять все возможные меры к сокращению длины программного комплекса, в первую очередь за счет сокращения длины кросспрограммы, а затем и модулей.

Способы сокращения длины программного комплекса или программы. В изложенной выше методике программирования на ПМК не предусматривалось получение программ минимальной длины. Здесь ставились другие цели: дисциплинировать процесс разработки программ, сократить общее время на их разработку, отладку, решение и облегчить повторное использование ранее разработанных программ. Поэтому в случае необходимости длина программного комплекса может быть, как правило, сокращена. Возможны следующие способы сокращения длины программ и программных комплексов.

1. Исключение участков программного комплекса или программы с повторяющимися командами и замена их подпрограммами. Для этого просматриваются все модули и кросспрограммы и отыскиваются в них участки с повторяющимися командами. При достаточной длине участка его следует заменить обращением к подпрограмме, выполняющей соответствующую последовательность команд, а подпрограмму поместить после последней команды программного комплекса. Обычно считается, что при двукратном повторении одинаковых участков их следует заменить подпрограммой при длине участка более пяти—шести шагов, а при трехкратном и более повторении, начиная с четырех шагов.

2. Использование команд с косвенной адресацией, что дает выигрыш в сокращении длины программы не только при использовании массивов, но и в других случаях, например при организации обычных повторений, когда в теле повторения необходимо производить вычитание или добавление единицы к значению какой-либо переменной. Например, записав одну команду *пп.*

КИП_i (где $i = 4, 5, 6$), можно обеспечить при каждом обращении к ней увеличение содержимого регистра на единицу. При выполнении этой команды происходит обращение к регистру i , после чего к его содержимому добавляется единица, а затем в регистр X заносится содержимое регистра, адрес которого находится в регистре i . При этом число, занесенное в регистр X, не используется. В частности, в модуле ЭРЛ — 6.1 команды 13.1 (1), 14. пX5 (i , 1) 15. + ($i + 1$) можно заменить одной командой 13.КпX5, выполняющей все действия, которые производились ранее командами 13—15. При этом содержимое регистра X после выполнения команды КпX5 не используется. В тех случаях, когда надо последовательно вычесть единицу, вместо регистров $i = 4, 5, 6$ можно использовать регистры $i = 0, 1, 2, 3$.

3. Использование в программном комплексе других, более коротких модулей и подпрограмм, если ограничения на использование последних несущественны для работы программного комплекса.

Длину модулей можно сократить, если входящие в них внешние подпрограммы поместить после команды останова, завершающего работу программного комплекса. В этом случае не придется обходить подпрограммы с помощью двухшаговых команд БП_{пп}, помещаемых перед началом каждой подпрограммы для перехода к выполнению других модулей. Тогда за счет размещения подпрограмм вне последовательности команд программного комплекса сокращается общая длина последнего на два шага для каждой вынесенной подпрограммы. В нашем примере, если подпрограмму БРВ, входящую в состав модуля МОК — 7.4, записать после команды С/П, то длина программного комплекса сократится на два шага и будет составлять 92 шага вместо 94.

4. Изменение кросспрограммы и модулей за счет применения нетрадиционных (эвристических) способов программирования. Это наиболее трудоемкий способ, требующий навыков и опыта в программировании. Но к нему приходится прибегать, когда не хватает места в памяти ПМК для размещения 1—2 команд программного комплекса и все другие способы сокращения длины программного комплекса исчерпаны.

В случае, если программный комплекс все же не удастся разместить в памяти ПМК, то придется изменить алгоритм вычислений путем дополнительного введения ручных операций, выполняемых пользователем ПМК. В частности, в нашем случае вычисление $\alpha = \lambda/\mu$ из-за ограниченного объема памяти ПМК производится вручную. В таких случаях усложнение решаемых с помощью ПМК задач возможно только за счет уменьшения степени автоматизации их решения и увеличения количества ручных операций. Если такая задача встречается нечасто, то это может быть приемлемым. Если же решение подобной задачи приходится выполнять систематически, то ее решение целесообразно выполнять на микроЭВМ, обладающей значительно большими памятью и бы-

стродействием. Решение ее на ПМК можно рассматривать как этап разработки программы на микроЭВМ, позволяющей проверить основные элементы алгоритма решения задачи.

Проверка корректности использования регистров адресуемой памяти ПМК в программном комплексе. Порядок проверки для модулей остается таким же, как и в случае использования подпрограмм в основной программе, рассмотренном в п. 7.2.6. Здесь следует отметить, что при сборке модулей в программный комплекс использование ранее выделенных регистров оказывается некорректным гораздо чаще. При этом необходимо (как и в случае подпрограмм) произвести перераспределение регистров памяти, выделенных для хранения переменных в кросспрограмме и модулях, а несовместные регистры заменить другими.

7.3.5. ЗАПИСЬ ПОЛНОГО ТЕКСТА ПРОГРАММНОГО КОМПЛЕКСА И ЕГО ОТЛАДКА

Составить программный комплекс — значит записать полный текст последовательности его команд в том виде, в котором он должен быть введен в память ПМК. Для этого следует переписать команды исходных модулей и кросспрограммы, дав им новые адреса в соответствии с их размещением в программном комплексе. В нашем случае последовательность команд программного комплекса будет иметь вид, представленный ниже (в программе 7.5). Составленный программный комплекс вводится в память ПМК в общем порядке (п. 5.3), а затем подлежит отладке. Отладка программного комплекса имеет свои особенности, которые заключаются в следующем.

1. При отладке программного комплекса прежде всего проверяется работоспособность модулей, входящих в его состав, поскольку при переадресации команд и изменениях в модулях, связанных с включением в программный комплекс, могут возникать ошибки. Для проверки каждого модуля после его последней команды, включенной в программный комплекс, в память ПМК временно заносится команда С/П. Далее, выбрав новый пусковой адрес командой БП_{пп} (где pp — пусковой адрес), произведем проверку работы модуля по тесту его основного алгоритма (команды В/О С/П в тесте заменяются командами БП_{пп} С/П в модуле). Если полученный результат соответствует приведенному в тесте, то модуль с новой адресацией команд, предусмотренной в программном комплексе, функционирует нормально. После этого вместо временно введенной команды С/П помещают прежнюю команду кросспрограммы. Таким же образом проверяются и внешние подпрограммы, работающие совместно с программным комплексом. Проверку функционирования модулей целесообразно начинать с проверки работы внешних программ.

2. После проверки функционирования модулей и внешних подпрограмм приступают к отладке программного комплекса в целом. Отладка производится по общим правилам (п. 5.4). При этом поиск ошибок производится только в кросспрограмме, поскольку до этого модули и внешние подпрограммы уже были проверены.

П р и м е ч а н и е. После того, как программный комплекс проверен и отлажен, при дальнейшем его использовании проверка отдельных модулей и внешних подпрограмм обычно не производится.

3. При отладочном выполнении программного комплекса (путем последовательного нажатия клавиш ПП) следует иметь в виду, что оба шага двухшаговых команд выполняются после одного нажатия клавиши ПП.

7.8.6. РАЗРАБОТКА ДОКУМЕНТАЦИИ НА ПРОГРАММНЫЙ КОМПЛЕКС

После выполнения всех ранее перечисленных этапов разработки программного комплекса по правилам, изложенным в п. 5.5, оформляется документация на программный комплекс:

ПРОГРАММА 7.5 — обслуживание. Расчет вероятности обслуживания заявок на телефонные переговоры при каналах с ограниченной надежностью.
Расчетные формулы:

$$P_{\text{обс}} = 1 - P_{\text{отк}},$$

где $P_{\text{отк}}$ — вероятность отказа в обслуживании заявки, определяемая по формуле Эрланга

$$P_{\text{отк}} = \frac{\alpha^K}{K!} \sum_{i=0}^K \frac{\alpha^i}{i!},$$

где α — приведенная плотность потока заявок, равная отношению интенсивности потока заявок λ к интенсивности потока обслуживания μ ; K — математическое ожидание готовых к использованию каналов ($K \leq n$),

$$K = E \left(\sum_{m=1}^n P_n^m m \right),$$

где E — целая часть от (...); n — число каналов, имеющихся в наличии ($n \geq 1$); P_n^m — вероятность события, когда из n имеющихся каналов окажутся пригодными к использованию ровно m каналов, определяемых по формуле Бернулли,

$$P_n^m = C_n^m K_{\text{г}}^m (1 - K_{\text{г}})^{n-m},$$

где $K_{\text{г}}$ — коэффициент готовности канала к работе.

О г р а н и ч е н и я:

1. Поток заявок и поток обслуживания — простейшие, промежутки времени между поступлениями заявок и временем их обслуживания распределены по экспонциальному закону.

2. $n \geq 1, R \geq 1$.
3. $1 \geq K_{\text{г}} > 0$.

```

алг  обслуживание
      арг R4 = n, R6 = Kг, RA = α
      рез RX = K, RY = Pобс
нач
      R9 = K, pr R1, R2, R3, R5, R7, R8
      BB R4 = n, R6 = Kг, RA = α
      HX B/O C/P
      MOK-7.4 (арг R4 = n, R6 = Kг, рез RX = K, pr R1, R2, R3, R7, R8)
      1 SUB MOK - 7.4
      00. 1      11. πX4    22. πX8    33. БП      44. πX1    55. ×
      01. πX6    12. xΠ1    23. 1      34. 57     45. πX2    56. B/O
      02. —       13. πΠ    24. +      35. πX1    46. ÷
      03. Fx ≠ 0  14. 35    25. xΠ7    36. πX2    47. πX6
      04. 32      15. πX3    26. KπX7   37. —      48. ×
      05. 0       16. ×      27. πX7    38. 1      49. ×
      06. xΠ8    17. πX8    28. 1      39. πX6    50. FL1
      07. πX4    18. +      29. —      40. —      51. 52
      08. xΠ3    19. xΠ8    30. БП    41. Fxy    52. FL2
      09. πX3    20. FL3    31. 33    42. xΠ7    53. 44
      10. xΠ2    21. 09    32. πX4    43. 1      54. xΠ7
      K = знач
      2 R9 = RX
      если K = 0
      3 IF RX = 0 ELSE 5
      то K = 1
      4 RX = 1
      все
      ∂РЛ = 6.1 (арг R3 = K, RA = α, рез RX = Pобс, pr R1, R2, R5)
      5 R3 = RX
      6 SUB ∂РЛ = 6.1
      62. 1      67. πX5    72. πX2    77. +
      63. xΠ1    68. ÷      73. +      78. xΠ5
      64. xΠ2    69. πX1    74. xΠ2    79. FL3
      65. xΠ5    70. ×      75. 1      80. 66
      66. πXA    71. xΠ1    76. πX5    81. 1
      если K = 0
      7 RX = R9
      8 IF RX = 0 ELSE 11
      то Pобс = Pобс * Kг
      9 RX = R4
      10 RX = RX * R6
      K = 1
      11 RX = 1
      все
      ВЫ K, Pобс
      12 STOP
кон
      86. πX9 (K)
      87. Fx = 0
      88. 93
      89. F. (Pобс)
      90. πX6 (R6, Rобс)
      91. × (Pобс * Kг)
      92. 1 (1, Pобс * Kг)
      93. С/П (K, Pобс)

```

ИНСТРУКЦИЯ

BB R4 = n, R6 = K_r , RA = α ; НЖ В/О С/П; ВЫ RX = K, RY = $P_{обс}$
ТЕСТ

- 1) BB R4 = n = 5; R6 = K_r = 0.8; RA = α = 2; НЖ В/О С/П
ВЫ RX = K = 3; RY = $P_{обс}$ = 7.894737 — 01 (3 мин)
- 2) BB R4 = n = 4; R6 = K_r = 1; RA = α = 0.25; НЖ В/О С/П
ВЫ RX = K = 4; RY = $P_{обс}$ = 9.998732 — 01 (25 с)
- 3) BB R4 = n = 1; R6 = K_r = 1; RA = α = 0.25; НЖ В/О С/П
ВЫ RX = K = 1; RY = $P_{обс}$ = 8. — 01 (15 с)

7.3.7. ПРИМЕР ВЫПОЛНЕНИЯ РАСЧЕТА ПО ДОКУМЕНТИРОВАННОМУ ПРОГРАММНОМУ КОМПЛЕКСУ

По разработанной документации выполним расчет математического ожидания числа каналов, готовых к использованию, и вероятности обслуживания телефонных переговоров с учетом надежности используемых каналов связи. Расчет произведем для значений $n = 1; 3; 5; 10$ при $K_r = 0.8; 0.9; 1.0$ и $\alpha = 0.5; 1.0; 2.0$. Строго выполняя последовательность действий, предусмотренных программой 7.5, получим табл. 7.1.

Таблица 7.1

Расчет числа пригодных к использованию каналов (с учетом их надежности) и вероятности обслуживания заявок на телефонные переговоры

n	K_r	α	K	$P_{обс}$	n	K_r	α	K	$P_{обс}$
1	0.8	0.5	1	5.3333336 — 01	5	0.8	0.5	3	9.873418 — 01
		1.0	4	— 01			1.0	3	9.375 — 01
		2.0	1	2.6666664 — 01			2.0	3	7.894737 — 01
	0.9	0.5	1	6.0000003 — 01		0.9	0.5	4	9.984202 — 01
		1.0	4.5	— 01			1.0	4	9.9846154 — 01
		2.0	1	2.9999997 — 01			2.0	4	9.047619 — 01
	1.0	0.5	1	6.6666667 — 01		1.0	0.5	5	9.998421 — 01
		1.0	5	— 01			1.0	5	9.969325 — 01
		2.0	1	3.333333 — 01			2.0	5	9.633028 — 01
3	0.8	0.5	2	9.230769 — 01	10	0.8	0.5	7	9.999991 — 01
		1.0	2	— 01			1.0	7	9.99927 — 01
		2.0	6	— 01			2.0	7	0.965591 — 01
	0.9	0.5	2	9.230769 — 01		0.9	0.5	8	9.999999 — 01
		1.0	2	— 01			1.0	8	9.999909 — 01
		2.0	6	— 01			2.0	8	9.991405 — 01
	1.0	0.5	3	9.873418 — 01		1.0	0.5	10	1.0
		1.0	3	9.375 — 01			1.0	10	9.999999 — 01
		2.0	3	7.894737 — 01			2.0	10	9.999618 — 01

Упражнение

7.3.1. По программе 7.5 рассчитать значения K и $P_{обс}$ для следующих значений:

n	K_r	α	n	K_r	α
5	0.5	0.5	10	0.6	1.0
5	0.7	0.5	10	0.95	2.0
10	0.5	0.5			

Глава 8. РАЗРАБОТКА ПРОГРАММНЫХ КОМПЛЕКСОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ МЕТОДОМ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ

Во всех предыдущих главах рассматривались вопросы использования ПМК для выполнения различных расчетов, алгоритмы которых включали соответствующие расчетные формулы. Однако ПМК (как и любая ЭВМ) может быть использован не только для выполнения расчетов по формулам, но и для проведения математических экспериментов на разработанных для этой цели математических моделях. Эти модели реализуются на ПМК с помощью соответствующих алгоритмов и программ, а результаты решения задачи получаются в процессе длительных испытаний указанных моделей, выполняемых на ПМК в автоматическом режиме, практически без непосредственного участия пользователя. Данная глава посвящена вопросам разработки таких моделей, результаты испытаний которых носят статистический характер, т. е. могут быть представлены в виде вероятности желаемого события или средними значениями его характеристик. Метод построения и испытания таких моделей получил название метода статистических испытаний. Дальнейшее описание указанного метода ориентировано на использование программируемых микрокалькуляторов типов МК-52 и МК-61, поскольку возможности МК-54 (а также его разновидностей БЗ-34 и МК-56) для реализации на них статистических моделей ограничены.

8.1. ПОНЯТИЕ О МЕТОДЕ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ

Разработка электронных схем ведется обычно по функциональным составным частям — каскадам. Каждый каскад состоит из активных (транзисторов, микросхем) и пассивных (резисторов, конденсаторов и др.) элементов. Приступая к разработке каскада, обычно рассчитывают значения параметров его пассивных элементов исходя из требуемых выходных параметров каскада, а затем собирают и испытывают его макет. В результате отладки макета окончательно устанавливают номинальные значения параметров всех входящих в него пассивных элементов.

Однако в условиях серийного производства все радиоэлементы разработанного каскада неизбежно будут иметь разброс параметров. При этом возникает вопрос, каковы будут значения выходных параметров каскада с учетом реальных разбросов значений параметров радиоэлементов. Для решения этой задачи часто рассчитывают каскад на наихудший случай разброса. Но при этом игнорируется статистический характер изменения значений параметров радиоэлементов. Это ведет к неоправданно большому ущербочинению допусков, к специальному отбору радиоэлементов, усложнению настройки и регулировки, снижению процента годных изделий, а иногда и к ошибочному выводу о невозможности

построения каскада (блока, элемента схемы) с требуемыми выходными параметрами.

Влияние разброса параметров радиоэлементов в принципе можно определить экспериментально на физической модели — макете каскада электронной схемы путем замены (перепайки) радиоэлементов с различными значениями параметров (но в пределах заданного допуска) и измерения каждый раз выходных параметров каскада. При большом количестве заменяемых радиоэлементов можно определить статистические характеристики выходных параметров. Однако такие испытания весьма трудоемки и могут быть выполнены в весьма ограниченном объеме для особо ответственных элементов схемы.

Но можно поступить и по-другому: перейти от физического эксперимента с перепайкой радиоэлементов к математическому. Можно вместо физической модели каскада (его макета) создать его математическую модель в виде программы для ПМК (или ЭВМ), позволяющую по различным входным значениям параметров радиоэлементов получить соответствующие значения выходных параметров. Отклонения значений входных параметров радиоэлементов формируют с помощью датчика случайных чисел; статистическую обработку результатов также включают в программу вычислений.

Рассмотренный пример является частным случаем общего метода численного решения задач при помощи моделирования случайных величин, называемым методом статистических испытаний, или методом Монте-Карло. Второе название метода происходит от города Монте-Карло, знаменитого своими игорными домами. Оно напоминает о том, что простейшим прибором для получения случайных величин является рулетка. Общая идея решения задач указанным методом заключается в следующем.

1. Выбираются по жребию значения случайных параметров на входе исследуемой системы.

2. Воспроизводится процесс функционирования этой системы при выбранном значении параметров на входе и определяются ее параметры на выходе. При этом и входные и выходные параметры системы рассматриваются как детерминированные. Такое однократное воспроизведение функционирования системы называется статистическим испытанием.

3. После многократного проведения статистических испытаний полученные результаты подвергаются статистической обработке с целью представления их в виде статистических характеристик (вероятности появления желаемого события, средние значения параметров, возможные отклонения значений параметров от среднего и др.).

Теоретическую основу метода статистических испытаний составляют предельные теоремы теории вероятностей, согласно которым при большом числе опытов частота появления события приближается к его вероятности (теорема Бернулли), а среднее

арифметическое значение случайной величины — к ее математическому ожиданию (теорема Чебышева) [6, 8]. Общий характер этих теорем обуславливает универсальность метода: он не связан практически ни с какими допущениями относительно исследуемых систем и позволяет решать достаточно сложные задачи при известных вероятностных характеристиках входных параметров системы и их взаимосвязях. Кроме того, для приближенного решения многих трудоемких детерминированных задач (вычисление кратных интегралов, численное интегрирование дифференциальных уравнений) можно построить искусственные вероятностные модели и также воспользоваться этим методом.

К числу достоинств метода статистических испытаний можно отнести:

- наглядную вероятностную трактовку;
- достаточно простую схему вычислений;
- малую чувствительность к отдельным ошибкам;
- простоту оценки точности получаемых результатов.

Вместе с тем рассматриваемому методу присущ и ряд недостатков, состоящих в необходимости:

знания закона изменений случайных значений параметров реальной системы и построения в статистической модели адекватного датчика случайных чисел;

большого количества статистических испытаний для получения приемлемой точности результатов.

Во многих случаях закон изменения случайных значений входных параметров системы известен лишь приблизительно или предположительно. Для более точного решения задачи часто приходится определять его путем физического эксперимента. Например, определять закон распределения фактических значений сопротивлений и емкостей от их номиналов, выпускаемых различными заводами-изготовителями. Тем не менее измерить значения сопротивлений нескольких сот резисторов гораздо проще, чем их впаивать в макет каскада и измерять при этом каждый раз его выходные параметры. Если же этот закон становится известным, то возникает задача генерирования последовательности случайных чисел по заданному закону их распределения, что требует разработки специальных подпрограмм, называемых датчиками случайных чисел (ДСЧ) или датчиками случайных величин (ДСВ).

Для получения достаточной точности результата требуется большое количество статистических испытаний (многие тысячи). Однако для получения точности результата порядка 10 % число испытаний может быть ограничено до 500—1000. Это позволяет для относительно несложных задач (3—5 входных параметров) реализовать метод статистических испытаний на ПМК. Программа для ПМК, реализующая этот метод, должна состоять из следующих основных частей:

1) датчика случайных чисел с требуемым законом распределения;

2) моделирующего алгоритма (программы), позволяющего по значениям входных параметров получить соответствующие им значения выходных параметров, при этом входные параметры определяются с помощью ДСЧ;

3) программы, осуществляющей прерывание статистических испытаний по усмотрению пользователя и в случае необходимости их продолжение без искажения результатов;

4) программы статистической обработки результатов серии испытаний: вычисления вероятности появления желаемого события, среднего арифметического значений выходных параметров и возможных отклонений от последнего;

5) программы оценки точности полученных результатов.

Решение задач на ПМК методом статистических испытаний занимает обычно от нескольких часов до нескольких суток из-за малого быстродействия ПМК. Поэтому оно может быть выполнено только при питании ПМК от сети. Длительность решения задачи при этом не имеет существенного значения. После запуска программы пользователь может заниматься другим делом по своему усмотрению. Он может вечером запустить программу, лечь спать, а утром получить результат. Но при этом обязательно должна быть обеспечена возможность в любое время по усмотрению пользователя прервать выполнение испытаний, оценить результаты уже проведенных и решить вопрос о необходимости их дальнейшего продолжения.

8.2. МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН

8.2.1. СЛУЧАЙНЫЕ ВЕЛИЧИНЫ И ИХ ХАРАКТЕРИСТИКИ

Понятие случайной величины. Говоря о случайной величине, нельзя сказать, какое значение она примет в данном конкретном случае, но можно назвать вероятности тех или иных ее значений; нельзя точно предсказать результат одного испытания, связанного с этой случайной величиной, но можно весьма надежно предсказать совокупность результатов большого числа испытаний. Чем больше испытаний (или, как говорят, чем больше статистика), тем точнее будут предсказания. Отсюда следует, что для того чтобы задать случайную величину, надо указать, какие значения она может принимать и каковы вероятности этих значений.

Случайная величина X может быть определена множеством ее возможных значений и вероятностей их появления:

$$\begin{pmatrix} x_1 & \dots & x_i & \dots & x_n \\ p_1 & \dots & p_i & \dots & p_n \end{pmatrix},$$

где $x_1 \dots x_i \dots x_n$ — возможные значения X ; $p_1 \dots p_i \dots p_n$ — соответствующие им вероятности.

Иными словами, p_i есть вероятность того, что X примет значение x_i , т. е. $P(X = x_i) = p_i$. Вероятности p_i должны удовлетворять требованиям

$$p_i \geq 0 \quad (i = 1, n); \quad \sum_{i=1}^n p_i = 1. \quad (8.1)$$

Приведенная матрица называется распределением случайной величины X . (Сумма $\sum p_i = 1$ распределяется между значениями x_i , отсюда название «распределение»).

Характеристики случайных величин. Математическое ожидание случайной величины X определяется как

$$M[X] = \sum_{i=1}^n x_i p_i. \quad (8.2)$$

При большом числе испытаний среднее арифметическое полученных значений $M^*[X]$ приближается к математическому ожиданию (или, как говорят, сходится по вероятности). Пусть произведено N независимых испытаний. При этом значение x_1 появилось m_1 раз, значение x_2 — m_2 раз, x_i — m_i раз и т. д. Тогда среднее арифметическое

$$\begin{aligned} M^*[X] &= \frac{x_1 m_1 + \dots + x_i m_i + \dots + x_n m_n}{N} = \\ &= x_1 \frac{m_1}{N} + \dots + x_i \frac{m_i}{N} + \dots + x_n \frac{m_n}{N} = \sum_{i=1}^n x_i \frac{m_i}{N}. \end{aligned}$$

Величина m_i/N является частотой (или статистической вероятностью) появления события $X = x_i$ и ее можно обозначить p^* . Тогда

$$M^*[X] = \sum_{i=1}^n x_i p_i^*. \quad (8.3)$$

Основные свойства математического ожидания:

$$\begin{aligned} M[X + c] &= M[X] + c; \\ M[X + Y] &= M[X] + M[Y], \end{aligned} \quad (8.4)$$

где c — неслучайная величина; X и Y — случайные величины.

Дисперсией $D[X]$ случайной величины X характеризуется разброс ее значений, т. е. возможные отклонения ее значений от математического ожидания:

$$D[X] = M[(M[X] - X)^2]. \quad (8.5)$$

Это выражение можно преобразовать в более удобное для вычислений на ПМК, используя свойства (8.4):

$$\begin{aligned} D[X] &= M[(M^2[X] - 2M[X] \cdot X + X^2)] = \\ &= M^2[X] - 2M[X] \cdot M[X] + M[X^2], \end{aligned}$$

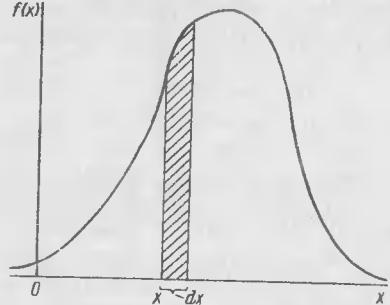


Рис. 8.1. График плотности распределения: $f(x) dx$ — элемент вероятности, значение которой равно заштрихованной площади, равной вероятности попадания случайной величины X в интервал dx , примыкающий к x .

торая описывает распределение явлений. При этом предполагается, что случайная величина является непрерывной, т. е. может принимать любые значения в заданном интервале, а не только фиксированные, как это имело место для дискретной случайной величины.

Интегральным законом распределения случайной величины X называется функция $F(x)$, (где x — некоторая текущая переменная) значения которой равны вероятности того, что случайная величина X не превзойдет x , т. е. $F(x) = P(X < x)$ (см. п. 8.2.3).

Дифференциальным законом распределения $f(x)$ (или плотностью распределения, плотностью вероятности) называется производная от интегрального закона распределения: $f(x) = F'(x)$ (рис. 8.1). Свойства дифференциального закона плотности распределения:

$$f(x) \geq 0;$$

$$\int_{-\infty}^{+\infty} f(x) dx = 1; \quad P\{\alpha < X < \beta\} = \int_{\alpha}^{\beta} f(x) dx. \quad (8.8)$$

Дифференциальная плотность распределения и интегральный закон распределения, иначе называемый функцией распределения, — это две различные формы представления одного и того же закона распределения. Закон распределения чаще всего задается в виде плотности распределения (дифференциального закона). Переход от плотности распределения к интегральному закону распределения осуществляется путем интегрирования:

$$F(x) = \int_{-\infty}^x f(x) dx.$$

откуда

$$D[X] = M[X^2] - M^2[X]. \quad (8.5a)$$

Основные свойства дисперсии:

$$D[X + c] = D[X]; \quad (8.6)$$

$$D[cX] = c^2 D[X],$$

где c — неслучайная величина.

Средним квадратическим отклонением случайной величины X называют корень квадратный из дисперсии:

$$\sigma[X] = \sqrt{D[X]}. \quad (8.7)$$

Законы распределения случайных величин. *Законом распределения случайной величины называется всякая функция, ко-вероятностей между ее значени-ями. При этом предполагается, что случайная величина является непрерывной, т. е. может принимать любые значения в заданном интервале, а не только фиксированные, как это имело место для дискретной случайной величины.*

Интегральным законом распределения случайной величины X называется функция $F(x)$, (где x — некоторая текущая переменная) значения которой равны вероятности того, что случайная величина X не превзойдет x , т. е. $F(x) = P(X < x)$ (см. п. 8.2.3).

Дифференциальным законом распределения $f(x)$ (или плотностью распределения, плотностью вероятности) называется производная от интегрального закона распределения: $f(x) = F'(x)$ (рис. 8.1). Свойства дифференциального закона плотности распределения:

При решении задач методом статистических испытаний наиболее часто используются равномерный, нормальный и экспоненциальный законы распределения.

Равномерный закон распределения. Если заранее известно, что значение случайной величины лежит в пределах интервала (α, β) , и в пределах его все значения случайной величины имеют одинаковую плотность распределения, то говорят, что они распределены по равномерному закону — закону равномерной плотности (рис. 8.2). Так как площадь, ограниченная кривой плотности распределения, равна единице [согласно (8.8)], т. е. $C(\beta - \alpha) = 1$, то для равномерного закона выражение для плотности вероятности можно записать так:

$$f(x) = \begin{cases} \frac{1}{\beta - \alpha} & \text{при } \alpha < x < \beta; \\ 0 & \text{при } x \leq \alpha \text{ или } x \geq \beta. \end{cases}$$

Равномерный закон распределения является наиболее случайным из всех других законов. Распределение значений случайных величин по этому закону соответствует распределению показаний идеальной игральной кости или рулетки.

Последовательность случайных чисел, распределенных по равномерному закону, используется в качестве исходной для формирования случайных величин, подчиняющихся другим законам распределения. В последнем случае в качестве границ интервала берутся значения $\beta = 1$ и $\alpha = 0$. Такая плотность распределения называется нормированной и записывается следующим образом:

$$\varphi(x) = \begin{cases} 1 & \text{при } 0 < x < 1; \\ 0 & \text{при } x \leq 0 \text{ или } x \geq 1. \end{cases}$$

Нормальный закон распределения, или закон Гаусса, является предельным законом, к которому приближаются все другие законы распределения при часто встречающихся типичных условиях. Он характеризуется плотностью вероятности (рис. 8.3). Максимальная ордината кривой, равная $1/(c\sqrt{2\pi})$, соответствует точке $x = m$. По мере удаления от точки m плотность распределения падает и при $x \rightarrow \pm\infty$ кривая асимптотически приближается к оси абсцисс. Для генерирования случайных величин, распределенных по нормальному закону, используют нормальное распределение с параметрами $m = 0$ и $c = 1$. Такое распределение называется нормированным. Его плотность

$$\varphi(x, 0, 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}. \quad (8.9)$$

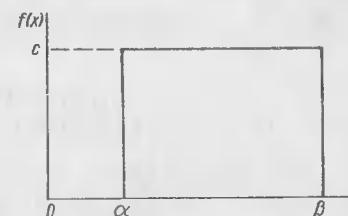


Рис. 8.2. Равномерный закон распределения. Плотность вероятности:

$$f(x) = \begin{cases} C & \text{при } \alpha < x < \beta; \\ 0 & \text{при } x \leq \alpha \text{ или } x \geq \beta \end{cases}$$

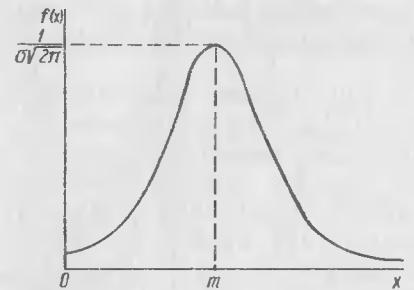


Рис. 8.3. Нормальный закон распределения. Плотность вероятности:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-m)^2/2\sigma^2},$$

где m — математическое ожидание случайной величины x ; σ — среднее квадратическое отклонение x от m .

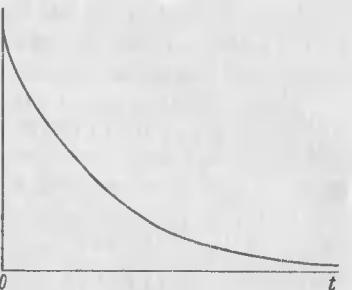


Рис. 8.4. Экспоненциальный закон распределения. Плотность вероятности: $f(t) = \lambda e^{-\lambda t}$ при $0 \leq t < \infty$; λ — средняя интенсивность потока заявок (количество заявок в единицу времени)

Экспоненциальный закон распределения широко используется для описания потока заявок в системах массового обслуживания. Для большинства часто встречающихся случаев предполагается, что поток заявок является простейшим, которому присущи следующие свойства:

стационарность, т. е. когда вероятность поступления k заявок за интервал времени (t_0, t) не зависит от t_0 , а зависит от k и t ;
ординарность, когда появление двух и более заявок в один и тот же момент времени практически невозможно;

отсутствие последействия, когда вероятность поступления k заявок за интервал времени (t_0, t) не зависит от того, сколько их поступило до наступления момента времени t_0 .

Случайный характер потока заявок может быть описан плотностью распределения интервалов времени между моментами поступления заявок (рис. 8.4). На указанном рисунке приведено выражение для плотности экспоненциального распределения, при котором максимальная вероятность поступления заявок будет иметь место при $t = 0$, а минимальная — при $t = \pm\infty$.

8.2.2. МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН С ЗАДАННЫМИ ЗАКОНОМ РАСПРЕДЕЛЕНИЯ

Для моделирования случайных величин на ПМК и ЭВМ с требуемыми законами распределения используются так называемые псевдослучайные числа, изменение значений которых от числа к числу подчиняется заданному закону распределения. Псевдослучайные числа формируются с помощью детерминированного алгоритма и, строго говоря, не являются случайными. Кроме недостатка в этом заключается и достоинство. Если последовательность «настоящих» случайных чисел повторить невозможно,

то все значения псевдослучайных чисел благодаря детерминированности алгоритма можно в точности повторить, а следовательно, и проверить окончательные результаты путем повторного решения на ПМК (ЭВМ), сопоставив их с результатами предыдущего решения.

В отличие от чисто случайных чисел псевдослучайным всегда свойственна периодичность. Это происходит потому, что количество цифр псевдослучайного числа ограничено и неизбежно наступает момент, когда все возможные сочетания цифр при данном алгоритме их формирования будут исчерпаны. Однако практически разрабатывают и используют такие алгоритмы формирования псевдослучайных чисел, неповторяющиеся фрагменты которых содержат тысячи и сотни тысяч различных чисел. Поэтому при хорошем алгоритме этот недостаток не является существенным. Датчики случайных чисел генерируют, как правило, псевдослучайные числа, равномерно распределенные в интервале $(0, 1)$. Они используются в качестве стандартных случайных чисел, из которых формируются случайные величины, распределенные по другим законам и/или с другими интервалами.

Случайные величины, равномерно распределенные в интервале $(0, 1)$, условимся в дальнейшем называть *случайными числами*. А говоря о случайных величинах, будем понимать под ними их значения, сформированные путем преобразования равномерно распределенных в интервале $(0, 1)$ псевдослучайных чисел. Получение очередного случайного псевдослучайного числа будем называть его *генерацией*, а генерацию случайного числа вместе с преобразованием его в случайную величину, распределенную по заданному закону, — *разыгрыванием* случайной величины.

8.2.3. ГЕНЕРАЦИЯ СЛУЧАЙНЫХ ЧИСЕЛ С РАВНОМЕРНЫМ ЗАКОНОМ РАСПРЕДЕЛЕНИЯ В ИНТЕРВАЛЕ $(0, 1)$

Обозначим r_i — случайное число, равномерно распределенное в интервале $(0, 1)$. Для генерации последовательности таких чисел используются рекуррентные способы вычисления последующего r_{i+1} числа из предыдущего r_i . Оно осуществляется с помощью функции преобразования

$$r_{i+1} = Q(r_i). \quad (8.10)$$

В настоящее время разработано большое количество датчиков случайных чисел, использующих различные формулы и алгоритмы преобразования. В качестве примера функции преобразования может быть использована рекуррентная формула

$$r_{i+1} = F(11r_i + \pi), \quad (8.10a)$$

где F — дробная часть числа.

В качестве r_i в данной функции может быть любое положительное число. Начальное значение $r_i = r_0$ задается пользователем.

Для каждого значения r_0 будет получена своя последовательность случайных чисел. Задавая различные значения r_0 , можно будет получить различные последовательности. Подпрограмма, реализующая датчик случайных чисел на МК-52 и МК-61 с использованием приведенной выше функции преобразования, будет выглядеть следующим образом:

ПОДПРОГРАММА 8.1 — ДСЧ. Генерация очередного случайного числа последовательности чисел, распределенных по равномерному закону в интервале $(0, 1)$. Исходная формула: $r_{i+1} = F(11r_i + \pi)$, где r_i — очередное случайное число; F — дробная часть числа [21]

```
алг ДСЧ
    арг RD =  $r_i$ 
    рез RX =  $r_{i+1}$ 
нач
    RD =  $r_{i+1}$ 
     $r_{i+1} = F(11r_i + \pi)$ 
```

```
1 RD = F(11 * RD + PI)
    00. пXD (RD)
    01. 1 (1, RD)
    02. 1 (11, RD)
    03. X (11 * RD)
    04. Fπ (PI, 11 * RD)
    05. + (11 * RD + PI)
    06. K {x} ( $r_{i+1}$ )
    07. xПD (RD =  $r_{i+1}$ )
2 RETURN
    08. В/О ( $r_{i+1}$ )
```

ВОЗВРАТ

2 RETURN

```
кон
TEST
BB RD =  $r = 0.5$ ; НЖ В/О С/П; ВЫ 6.415926 —01 (3 с)
НЖ В/О С/П; ВЫ 1.99111 —01 (3 с)
НЖ В/О С/П; ВЫ 3.318136 —01 (3 с)
```

Команда генерации случайного числа. В МК-52 и МК-61 имеется встроенный датчик случайных чисел. Генерация очередного случайного числа в интервале $(0, 1)$ осуществляется с помощью команды КСЧ. Шаблон команды:

знач = случайное число

```
A RX = RND
    a + 0. КСЧ
```

В правой части БЕЙСИК-команды записывается функция RND. В качестве начального значения случайного числа при включении ПМК всегда берется нулевое значение. При этом первым случайнм числом будет $r_1 = 4.04067 - 01$. Для сброса встроенного датчика случайных чисел на нуль без выключения ПМК необходимо обнулить все регистры стека и установить ДСЧ в начальное состояние путем нажатия клавиш НЖ Сх В! В! В! КЗН.

Преупреждение. Для решения задач методом статистического моделирования командой КСЧ лучше не пользоваться. Последовательное нажатие клавиши КСЧ вызывает генерацию только 196 неповторяющихся чисел (сначала генерируются 53 неповторяющихся числа, а затем — повторяющиеся группы по 143 числа в каждой). При этом распределение генерируемых чисел нельзя назвать равномерным.

Кроме того, значения выдаваемых случайных чисел зависят от содержимого стековых регистров и в различных программах будут различными. Использование некоторых команд с префиксом К вызывает повторную генерацию ранее выданных чисел и даже возврат ДСЧ в начальное состояние. При необходимости использования команды КСЧ рекомендуется:

1. Исключить в программе команды с использованием префикса К, особенно КЗН, КΔ, К∇, КΦ, Ктах, КИНВ, К, , → 0, К | |.

2. Перед командой КСЧ поместить в регистр Х предыдущее случайное число, полученное от предшествующего выполнения команды КСЧ, а после команды поместить полученное случайное число в регистр, выделенный для его хранения, т. е. вместо одной команды КСЧ рекомендуется использовать серию из трех команд: а+0. пXi а+1. КСЧ а+2. хPi, где i — адрес регистра, который выделяется для хранения случайного числа.

Оценка качества датчиков случайных чисел. Поскольку случайные числа являются источниками (исходными данными) для получения других случайных величин, то к датчикам случайных чисел должны быть жесткие требования. Иначе при их невыполнении сформированные случайные величины не будут соответствовать заданным законам распределения. Качество датчиков случайных чисел может быть оценено:

1) функцией преобразования r_i в r_{i+1} ;

2) количеством генерируемых неповторяющихся чисел $N_{\text{нп}}$;

3) коэффициентом неравномерности распределения генерируемых случайных чисел в процентах:

$$K_h = \frac{1}{N} \sqrt{\frac{\sum_{j=1}^n (N_j - \bar{N})^2}{n}} \cdot 100, \quad (8.11)$$

где N_j — количество случайных чисел, попадающих в j -й интервал гистограммы; n — число интервалов гистограммы; $\bar{N} = N_{\text{общ}}/n$ — среднее количество случайных чисел, попадающих в интервал гистограммы; $N_{\text{общ}}$ — общее количество случайных чисел, выданное ДСЧ; h — длина частичного интервала гистограммы.

Значения параметров основных датчиков случайных чисел, рекомендуемых для МК-52, -61, приведены в табл. 8.1. Они соответствуют значениям начальных (стартовых) чисел ДСЧ r_0 , указанным в таблице. В случае других значений r_0 параметры ДСЧ могут сильно отличаться от указанных. Особенно критичен к изменениям ДСЧ № 1 ($r_{i+1} = F(37r_i)$): при замене значения

4) временем генерации одного случайного числа t_g , с;

5) требуемыми ресурсами памяти ПМК для реализации датчика случайных чисел: программной памяти, определяемой длиной подпрограммы ДСЧ; регистровой памяти, определяемой числом адресуемых регистров, необходимых для выполнения подпрограммы ДСЧ.

$R_0 = 0.1234567$ на $r_0 = 0.5$ этот ДСЧ вырождается, т. е. начинает выдавать каждый раз одно и то же число $r_{i+1} = r_i = r_0 = 0.5$.

Таблица 8.1

Характеристики датчиков случайных чисел, распределенных по равномерному закону в интервале $(0, 1)$ и реализованных на МК-52 и МК-61

Формула преобразования	Количество неповторяющихся чисел $N_{\text{НП}}$	Коэффициент неравномерности K_n , %	Время генерации одного числа t_g , с	Требуемые ресурсы		Источник
				ПрП ячеек, шаги	регистров	
$r_{i+1} = F(37 r_i);$ $r_0 = 0.1234567$	>42301	6.9	2	7	1	[13]
$r_{i+1} = F(11r_i + \pi);$ $r_0 = 0.5$	>8000	11.1	3	9	1	[21]
$r_{i+1} = F(r_i/z_i + \pi);$ $z_{i+1} = z_i + 10^{-8};$ $r_0 = 0, z_0 = 0.011$	>8.9·10 ⁷	7.7	3.5	16	2	[21]

Примечания. 1. r_i — предыдущее значение случайного числа, r_{i+1} — последующее, r_0 — начальное.
2. F — дробная часть от выражения в скобках.
3. Коэффициент неравномерности K_n определен при общем количестве выдаваемых случайных чисел $N_{\text{общ}} = 1000$ при 10 интервалах гистограммы.

Для определения коэффициента неравномерности K_n выбрана наиболее часто используемая на ПМК первая тысяча генерируемых чисел, для которой режим ДСЧ еще не является установленным.

Подпрограммы для ДСЧ табл. 8.1 даны в решениях к упр. 8.2.1.

Выбор датчика случайных чисел для использования в статистических моделях производится исходя из наличия свободных ресурсов программной памяти ПМК, незанятых регистров адресуемой памяти и требований к точности вычислений. Меньшее значение имеет время генерации одного числа, поскольку программы их преобразования в случайные величины с последующей обработкой требуют для своего выполнения на порядок большего времени.

8.2.4. РАЗЫГРЫВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН С ЗАДАННЫМ ЗАКОНОМ РАСПРЕДЕЛЕНИЯ

Как уже указывалось, разыгрывание случайной величины представляет собой процесс преобразования случайного числа, равномерно распределенного в интервале $(0, 1)$, в случайную величину, распределенную по заданному закону. Для реализации указан-

ного процесса необходимо представить законы распределения случайного числа и случайной величины в интегральной форме:

$$F(x) = \int_{-\infty}^{+\infty} f(x) dx.$$

Интегрируя выражение для плотности распределения равномерного закона:

$$\varphi(x) = \begin{cases} 1 & \text{при } 0 < x < 1; \\ 0 & \text{при } x \leq 0 \text{ и } x \geq 1 \end{cases}$$

и заменив переменную x на переменную r , получим выражение для интегрального закона равномерного распределения (рис. 8.5, кривая 1). На этот же график в том же масштабе должна быть нанесена кривая интегрального закона другого распределения, которому должны подчиняться разыгрываемые случайные величины (рис. 8.5, кривая 2). Каждая ордината графика функции распределения показывает значение вероятности того, что случайное число R окажется меньше r , т. е. $F(r) = P(R < r)$. Особенность этой функции: абсцисса равна ординате ($F(r) = r$).

Определив случайное число r , с помощью функций распределения $F(r)$ и $F(x)$ легко можно определить случайную величину X , удовлетворяющую последнему закону распределения $F(x)$, как это показано на рис. 8.5.

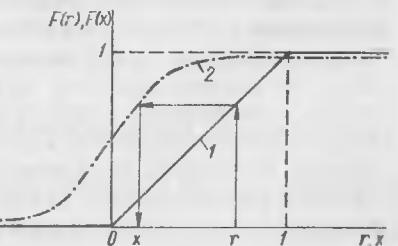
Если $F(r)$ и $F(x)$ заданы аналитически, то легко можно получить формулы преобразования значений r в x . В случаях, когда плотность распределения $f(x)$ не интегрируется и точное выражение для интегрального закона распределения получить не представляется возможным, используют приближенные выражения для интегрального закона распределения, случайные величины которого подлежат разыгрышу.

Если $F(x)$ представляет собой интегральную функцию равномерного распределения в интервале (α, β) , то, используя изложенную методику, можно получить функцию преобразования для

Рис. 8.5. Преобразование случайного числа r в случайную величину x : 1 — интегральная функция распределения по равномерному закону в интервале $(0, 1)$:

$$F(r) = \begin{cases} 0 & \text{при } r \leq 0; \\ r & \text{при } 0 < r < 1; \\ 1 & \text{при } r \geq 1; \end{cases}$$

2 — интегральная функция с заданным законом распределения



Характеристики датчиков нормированных нормально распределенных величин, реализуемых на МК-52 и МК-61

Формула преобразования	Время разыгрывания одногого числа	Требуемые ресурсы		Источник
		ПрП ячеек, шагов	регистров	
$n(0,1) = \text{sign}(2r-1) \sqrt{-\frac{\pi}{2} \ln(4r(1-r))},$ где $\text{sign } x = \begin{cases} 1 & x > 0, \\ 0 & x = 0, \\ -1 & x < 0; \end{cases}$	7 с	21	1	[21]
$n(0,1) = \sin(2\pi r_i) \sqrt{2 \ln(1/r_{i+1})};$	7 с	17	1	[13]
$n(0,1) = \sum_{i=1}^{12} r_i - 6;$	37 с	17	3	[2,8]
$n(0,1) = \sqrt{2} \sum_{i=1}^6 r_i - 3$	20 с	19	3	[2,8]

П р и м е ч а н и я: 1. Функция $\text{sign}(x)$ реализуется на ПМК с помощью команды КЗН.

2. Третья и четвертая формулы преобразования основаны на предельной теореме теории вероятностей, на основе которой при неограниченном возрастании числа случайных слагаемых в суммах, распределение значений сумм подчиняется нормальному закону. Формула 3 дает лучшее приближение к нормальному закону, чем формула 4.

3. Время разыгрывания и длина программы приведены без учета времени генерации и длины подпрограммы датчика случайных чисел.

участка программной памяти, длины программы и времени разыгрывания случайной величины.

Рассмотрим пример подпрограммы, реализующей на МК-52 и МК-61 датчик нормированной нормально распределенной случайной величины, по формуле преобразования 1, приведенной в табл. 8.2, и датчик случайных чисел ДСЧ1 — подпрограмма 8.1.

ПОДПРОГРАММА 8.2 — Гаусс. Датчик нормированных нормально распределенных величин [21].

И с х о д и н ы е д а н н ы е: r_0 — пусковое число ДСЧ или r — предыдущее случайное число, выданное ДСЧ.

Р е з у л ь т а т ы: $n(0,1)$ — нормированная нормально распределенная случайная величина с математическим ожиданием $m = 0$ и дисперсией $D = \sigma^2 = 1$; r — очередное случайное число для запуска ДСЧ.

$$r(\alpha, \beta) = \alpha + (\beta - \alpha) r(0, 1),$$

где $r(0, 1)$ — случайное число, равномерно распределенное в интервале $(0, 1)$; $r(\alpha, \beta)$ — случайная величина, равномерно распределенная в интервале (α, β) .

8.2.5. РАЗЫГРЫВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН, РАСПРЕДЕЛЕННЫХ ПО НОРМАЛЬНОМУ ЗАКОНУ

Интеграл от плотности вероятности, распределенной по нормальному закону, не выражается через элементарные функции. Поэтому интегральная функция распределения по нормальному закону может быть представлена лишь в виде графика, таблицы или приближенной формулы. Отсюда следует, что формула преобразования случайного числа в нормально распределенную случайную величину может быть только приближенной.

Разыгрывание нормальной случайной величины производится в два этапа. Сначала разыгрывается нормированная нормально распределенная случайная величина с математическим ожиданием $m = 0$ и дисперсией $D = \sigma^2 = 1$, а затем полученное значение случайной величины $n(0,1)$ преобразуется в ненормированное $n(m, D)$ с математическим ожиданием $m \neq 0$ и дисперсией $D = \sigma^2 \neq 1$.

Разыгрывание нормированной нормально распределенной величины производится с помощью одной из приближенных формул преобразования, приведенных в табл. 8.2. В этой таблице указаны также основные характеристики датчиков нормированных нормально распределенных величин, реализуемых на МК-52 и МК-61. Время разыгрывания и требуемые ресурсы указаны в ней без соответствующих значений датчика случайных чисел, входящего в качестве подпрограммы в датчик случайных величин. Число неповторяющихся значений случайных величин и их отклонение от нормального закона определяются в основном параметрами выбранного датчика случайных чисел.

Все формулы преобразования многократно проверены и при идеальном датчике случайных чисел разыгрываемые случайные величины всегда будут распределены по закону, достаточно близкому к нормированному нормальному. Количество неповторяющихся случайных величин также определяется выбранным датчиком случайных чисел. Однако другие характеристики разыгрываемой случайной величины (длина программы ПМК, осуществляющей разыгрывание, время ее выполнения) имеют существенное значение при выборе программы преобразования случайного числа в случайную величину. Выбор того или иного датчика случайной величины определяется исходя из наличия свободного

Расчетные формулы: $r_{i+1} = F(11r_i + \pi) = r$ (подпрограмма 8.1 — ДСЧ); $n(0,1) = \text{sign}(2r - 1) \sqrt{-\frac{\pi}{2} \ln(4r * (1 - r))}$

алг Гаусс
арг RD = r
рез RX = $n(0,1)$, RD = $r_{i+1} = r$

нач ДСЧ1 (арг RD = r_i рез RD = r)
1 SUB ДСЧ1

00. пXD 02. 1 04. Fπ 06. K {x}
01. 1 03. × 05. + 07. xΠD

знач = sign(2r - 1) $\sqrt{(-\pi/2) \ln(4r(1 - r))}$
2 RX = (SIGN(2 * RD - 1)) * SQR((-P1/2) *
LN(4 * (1 - RD)))
08. 4 (4, RD)
09. × (4 * RD)
10. 1 (1, 4 * RD)
11. πXD (RD, 1, 4 * RD)
12. — (1 — RD, 4 * RD)
13. × (4 * RD * (1 — RD))
14. Fln (LN(4 * RD * (1 — RD)))
15. Fπ (PI, 14, RX)
16. /—/ (-PI, 14, RX)
17. 2 (2, -PI, 14, RX)
18. ÷ (-PI/2, 14, RX)
19. × ((-PI/2 * 14, RX))
20. F √ (SQR(19, RX))
21. πXD (RD, 20, RX)
22. 2 (2, RD, 20, RX)
23. × (2 * RD, 20, RX)
24. 1 (1, 2 * RD, 20, RX)
25. — (2 * RD — 1, 20, RX)
26. K3H (SIGN(25, RX), 20RX)
27. × (n(0, 1))

ВОЗВРАТ

3 RETURN

28. B/O (n(0, 1))

кон
ТЕСТ
BB RD = $r_0 = 0.5$; НЖ В/О С/П; ВЫ 3.6236256 —01 (12 с)
НЖ В/О С/П; ВЫ -8.4040436 —01 (12 с)
НЖ В/О С/П; ВЫ -4.3429766 —01 (12 с)

Приведенная подпрограмма хороша тем, что в ней отсутствуют тригонометрические функции. Это позволяет очень удобно организовать прерывание выполняемой программы по усмотрению пользователя. Подпрограмма обеспечивает приемлемое время разыгрывания случайной величины.

Получение ненормированной нормально распределенной величины из нормированной осуществляется по формуле

$$n(m, \sigma) = m + \sigma * n(0, 1),$$

где $n(m, \sigma)$ — ненормированная случайная величина с математическим ожиданием m и дисперсией σ^2 ; $n(0, 1)$ — нормированная случайная величина.

8.2.6. РАЗЫГРЫВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН, РАСПРЕДЕЛЕННЫХ ПО ЭКСПОНЕНЦИАЛЬНОМУ ЗАКОНУ

Определим формулу преобразования случайного числа в случайную величину, распределение которой подчиняется экспоненциальному закону, описываемому плотностью распределения в интервале $(0, \infty)$ (см. рис. 8.4):

$$f(t) = \lambda e^{-\lambda t}.$$

Для того чтобы получить формулу преобразования для экспоненциального закона, необходимо этот закон представить в интегральной форме, проинтегрировав его плотность распределения

$$F(t) = \int_{-\infty}^t f(t) dt = \int_0^t \lambda e^{-\lambda t} dt = \lambda \left(-\frac{1}{\lambda} \right) [e^{-\lambda t}]_0^t = 1 - e^{-\lambda t}.$$

Абсцисса интегральной функции распределения для равномерного закона в интервале $(0, 1)$ равна ее ординате, т. е. $F(r) = r$ (рис. 8.5). Тогда, приравнивая $F(t) = F(r) = r$, получим

$$1 - e^{-\lambda t} = r.$$

Логарифмируя обе части этого равенства, придем к результату — формуле преобразования случайного числа в случайную величину, распределенную по экспоненциальному закону

$$t = -\frac{1}{\lambda} \ln(1 - r).$$

Так как значения $1 - r$ и r распределены по одному и тому же равномерному закону в том же интервале $(0, 1)$ (меняется лишь начальное значение последовательности, точка отсчета), то $1 - r$ можно заменить на r и получить окончательное выражение для формулы преобразования

$$t = -\frac{1}{\lambda} \ln r.$$

Подпрограмма датчика случайных величин, распределенных по экспоненциальному закону, представлена ниже:

ПОДПРОГРАММА 8.3 — экспонента. Генерация случайной величины, подчиняющейся экспоненциальному закону распределения.

Исходные данные: r_0 — пусковое число ДСЧ; r_i — предыдущее случайное число, выданное ДСЧ; λ — интенсивность потока заявок (число заявок в единицу времени).

Результаты: t — интервал времени между поступлением предыдущей и последующей заявок; r — очередное случайное число для запуска ДСЧ.

алг экспонента
арг RC = λ , RD = r_i
рез RX = t , RD = r

нач
ДСЧ (апр RD = r_i рез RD = r)
1 SUB DC4
00. пXD 02. 1 04. Fπ 06. K {x}
01. 1 03. X 05. + 07. xPID
2 RX = (-1/RC) * LN (RX)
08. F ln (LN (r))
09. пXC (RC, LN (r))
10. F1/x (1/RC, LN (r))
11. /-/-1/RC, LN (r))
12. X ((-1/RC) * LN (r))

ВОЗВРАТ
кон
ТЕСТ
3 RETURN 13. В/О (t)

BB RC = $\lambda = 10$, RD = $r_0 = 0.002$; НЖ В/О С/П; ВЫ 1.8103762 —01 (6 с)
НЖ В/О С/П; ВЫ 6.0694 —03 (6 с)
НЖ В/О С/П; ВЫ 7.055924 —02 (6 с)

Все рассмотренные подпрограммы датчиков случайных чисел и величин разработаны по известным формулам, испытаны и проверены длительной эксплуатацией на ПМК и ЭВМ [13, 21], поэтому соответствие получаемых чисел заданным законам распределения сомнений не вызывает. Когда закон распределения исходных данных к решаемой задаче известен, то можно воспользоваться соответствующим датчиком случайных величин. Если же закон распределения исходных данных неизвестен, то приходится определить его экспериментально. Как это делается, достаточно хорошо и доступно описано в книге Е. С. Вентцель ([6], гл. 7), а также в других многочисленных руководствах по математической статистике и теории вероятностей [1, 8].

Для установления факта соответствия распределения опытных данных нормальному закону можно воспользоваться правилом «трех сигм», которое гласит, что при нормальном законе распределения максимальное отклонение случайной величины от среднего ее значения с вероятностью 0,997 не превышает трех значений среднего квадратического отклонения σ . Если распределение изучаемой случайной величины неизвестно, но условие, приведенное в указанном правиле, выполняется, то есть все основания полагать, что изучаемая величина распределена нормально. В противном случае она распределена не нормально [8].

Упражнения

8.2.1. Составить программы датчиков случайных чисел с использованием следующих функций преобразования:

- 1) $r_{i+1} = F(37r_i)$, $r_0 = 0.1234567$;
- 2) $r_{i+1} = F(r_i/z_i + \pi)$; $z_{i+1} = z_i + 10^{-8}$, $z_0 = 0.011$, $r_i = 0$.

8.3. ОБРАБОТКА И ВЫДАЧА РЕЗУЛЬТАТОВ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ

8.3.1. ОБРАБОТКА РЕЗУЛЬТАТОВ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ

Процесс проведения статистических испытаний может продолжаться неограниченно долго, поэтому должна быть предусмотрена возможность его прекращения после окончания любого статистического испытания. Это в свою очередь требует проведения статистической обработки результатов всех предшествующих испытаний по окончании каждого очередного испытания. При этом с целью сокращения времени каждого испытания в процессе выполнения последнего обработка подвергаются лишь те данные, которые невозможно выполнить после окончания всей серии испытаний.

В зависимости от условия решаемой задачи статистическая обработка результатов по окончании как каждого испытания, так и всей серии испытаний может быть двух типов, предусматривающих:

вычисление статистической вероятности желаемого события;
вычисление средних значений и дисперсий исследуемых параметров, дискретные значения которых вычисляются при каждом испытании.

Вычисление статистической вероятности желаемого события. Пусть N общее количество проведенных статистических испытаний, N_z — количество испытаний с благоприятным исходом. Тогда статистическая вероятность появления события будет определена, как

$$p_z^* = \frac{N_z}{N}.$$

Определение значений N и N_z производится при каждом статистическом испытании, для чего в программе организуются счетчики общего числа испытаний N и числа испытаний с благоприятным исходом N_z . Статистическая вероятность p_z^* может быть вычислена при этом после окончания всей серии испытаний.

Вычисление среднего значения и дисперсии. Пусть z — значение некоторой величины, полученной при i -м статистическом испытании (результат i -го испытания). Тогда ее среднее значение при N испытаниях

$$M[z]_N = \frac{1}{N} \sum_{i=1}^N z_i = \sum_{i=1}^N \frac{z_i}{N}.$$

Среднее значение $M[z]_i$ должно вычисляться в конце каждого i -го испытания. При этом используется значение $M[z]_{i-1}$, полученное после выполнения $i-1$ предыдущих испытаний. Тогда вычисление среднего значения за N испытаний может быть выполнено с помощью рекуррентной формулы

$$M[z]_N = M[z]_{N-1} * (N-1)/N + z_N/N.$$

Помимо среднего значения необходимо также определить и дисперсию $D[z]_N$, т. е. возможный разброс отклонений определяемой величины относительно среднего значения. При каждом статистическом испытании для этого вычисляется квадрат полученного результата z^2 , а затем средний квадрат за все N проведенных испытаний по аналогичной рекуррентной формуле

$$M[z^2]_N = M[z^2]_{N-1} * (N - 1)/N + z_N^2/N.$$

Полученные значения $M[z]_N$ и $M[z^2]_N$ используют для вычисления значения дисперсии $D[z]$ и среднего квадратического отклонения σ_N . Это можно сделать по завершении всей серии статистических испытаний. Тогда, используя (8.5а) и (8.7), можно записать выражения для дисперсии $D[z]$ и среднего квадратического отклонения σ :

$$D[z] = M[z^2] - M^2[z];$$

$$\sigma_z = \sqrt{D[z]}.$$

Ниже приведена подпрограмма вычисления $M[z]$ и $M[z^2]$, осуществляемая после выполнения каждого испытания, можно реализовать на ПМК с помощью подпрограммы:

ПОДПРОГРАММА 8.4 — среднее. Вычисление среднего арифметического и среднего квадрата случайной величины z .

Исходные данные:

$N - 1$ — количество предшествующих выполненных уже статистических испытаний, не считая текущего, выполняемого;

$M[z]_{N-1}$, $M[z^2]_{N-1}$ — среднее значение z и z^2 за $N - 1$ предшествующих испытаний, не включая последнее.

Результаты:

N — количество произведенных статистических испытаний, включая и последнее;

$M[z]_N$, $M[z^2]_N$ — среднее значения z и z^2 за N испытаний, включая и последнее.

Расчетные формулы:

$$M[z]_N = M[z]_{N-1} * (N - 1)/N + z_N/N;$$

$$M[z^2]_N = M[z^2]_{N-1} * (N - 1)/N + z_N^2/N.$$

алг среднее
апр $R5 = N - 1$, $R6 = M[z]_{N-1}$, $R7 = M[z^2]_{N-1}$ $RX = z$
рез $R5 = N$, $R6 = M[z]_N$, $R7 = M[z^2]_N$
 $R8 = z$, $R9 = (N - 1)/N = N_1$
знач = z

$$1 R8 = RX \\ 00. x\Pi8 (R8 = z)$$

$$N_1 = (N - 1)/N$$

$$2 R9 = R5/(R5 + 1) \\ 01. \Pi X5 (R5) \\ 02. \Pi X5 (R5, R5) \\ 03. 1 (1, R5, R5) \\ 04. + (R5 + 1, R5) \\ 05. x\Pi5 (R5 = N + 1) \\ 06. \div (R5/(R5 + 1)) \\ 07. x\Pi9 (R9 = N_1)$$

1 $M[z]_N = M[z]_{N-1} * N_1 + z_N/N$
 $3 R6 = R9 * R6 + R8/R5$
 08. $\Pi X6 (R6, R9)$
 09. $\times (R9 * R6)$
 10. $\Pi X8 (R8, R9 * R6)$
 11. $\Pi X5 (R5, R8, R9 * R6)$
 12. $\div (R8/R5, R9 * R6)$
 13. $+$ ($R9 * R6 + R8/R5$)
 14. $x\Pi6 (R6 = M[z]_N)$

2 $M[z^2]_N = M[z^2]_{N-1} * N_1 + z_N^2/N$
 $4 R7 = R9 * R7 + R8 * * 2/R5$
 15. $\Pi X9 (R9)$
 16. $\Pi X7 (R7, R9)$
 17. $\times (R9 * R7)$
 18. $\Pi X8 (R8, R7 * R9)$
 19. $Fx^2 (R8 * * 2, R7 * R9)$
 20. $\Pi X5 (R5, R8 * * 2, R7 * R9)$
 21. $\div (R8 * * 2/R5, R7 * R8)$
 22. $+$ ($R7 * R9 + R8 * * 2/R5$)
 23. $x\Pi7 (R7 = M[z^2]_N)$

ВОЗВРАТ

5 RETURN

24. B/O ($M[z^2]_N$)

кон
тест

BB $R5 = N - 1 = 0$, $R6 = M[z]_{N-1} = 0$, $R7 = M[z^2]_{N-1} = 0$, $RX = z = 2$
 НЖ B/O C/P; ВЫ $R5 = N = 1$, $R6 = M[z]_N = 2$, $R7 = M[z^2]_N = 4$ (5 c)
 BB $z = 3$; НЖ B/O C/P; ВЫ $R5 = N = 2$, $R6 = M[z]_N = 2.5$, $R7 = M[z^2]_N = 6.5$

8.3.2. ВЫДАЧА РЕЗУЛЬТАТОВ РЕШЕНИЯ

Если время решения обычных задач на ПМК измеряется минутами, то время решения задач методом статистических испытаний может продолжаться от нескольких часов до нескольких суток в зависимости от того, какую точность результатов требуется получить. Поэтому решение задачи на ПМК методом статистических испытаний может быть получено только при питании ПМК от электрической сети. Желательно, чтобы автономное питание микрокалькулятора было подключено через диод [3], что позволит продолжать решение задачи при значительных (2—3 ч) перерывах электропитания от сети.

В принципе, чем дольше решается задача и больше производится статистических испытаний, тем точнее будут полученные результаты. Но так можно решать задачу до бесконечности. Кроме того, при длительном выполнении программы пользователь ПМК теряет всякую уверенность в том, что она правильно работает. Возникает потребность в любой момент приостановить выполнение программы, убедиться в правильности ее работы, определить количество уже произведенных статистических испытаний, оценить точность полученных результатов и, если она окажется недостаточной, продолжить проведение испытаний с момента останова программы.

В п. 5.2 отмечалось, что выполнение любой программы может быть приостановлено путем нажатия клавиши С/П. После ее нажатия останов будет произведен перед очередной командой выполняемой программы. Это может быть любая команда программы, поскольку пользователь не может знать, какая команда программы в данный момент выполняется. В том случае, когда задача решается методом статистических испытаний, останов должен быть произведен в одном и том же месте программы, после выполнения одной и той же команды. Выбор места останова определяется следующими условиями:

1. До выполнения команды останова должно быть полностью закончено предыдущее статистическое испытание и не начато последующее.

2. До выполнения команды останова должен быть произведен подсчет количества выполненных статистических испытаний и исходя из него определены результаты всей серии испытаний, пригодные для дальнейшей обработки.

При останове после нажатия клавиши С/П перечисленные условия, естественно, не могут быть выполнены, поскольку останов будет произведен в случайном месте программы. Поэтому для выполнения указанных условий требуется включение в состав основной программы специальной подпрограммы или участка программы, предусматривающей возможность останова только после завершения уже начатого испытания. Такой останов может быть выполнен:

- с помощью счетчика заданного числа испытаний;
- с помощью переключателя Р — ГРД — Г.

Останов с помощью счетчика заданного числа испытаний. Перед выполнением каждой серии статистических испытаний в один из регистров памяти вводится заданное число испытаний N_3 . После завершения каждого испытания производится сравнение показаний счетчика проведенных N и заданных N_3 испытаний. При их совпадении производится окончательная обработка результатов с выдачей статистических значений вероятности p^* или среднего арифметического m^* и дисперсии D^* за N_3 произведенных испытаний, после чего производится останов вычислений. Для продолжения статистических испытаний необходимо будет ввести новое значение $N_3 > N$ и запустить программу на продолжение решения, нажав клавишу С/П. Достоинством данного способа является его универсальность: он пригоден для любых программ статистических испытаний. Однако этот способ не позволяет выполнять останов в любое время по усмотрению пользователя: надо дожидаться окончания заданного числа испытаний.

Останов с помощью переключателя Р — ГРД — Г. Перед запуском программы переключатель Р—ГРД—Г должен быть поставлен в положение Р (радианы) или Г (градусы), при этом обеспечивается непрерывная работа программы от одного ста-

тистического испытания к другому. Если во время работы программы переключатель Р—ГРД—Г поставить в положение ГРД (грады), то перед началом следующего статистического испытания производится останов вычислений. Для продолжения испытаний следует снова перевести переключатель Р—ГРД—Г в положение Р или Г, а затем нажать клавишу С/П. Если же указанный переключатель остановить в положении ГРД и запустить программу нажатием клавиши С/П, то будет выполнено только одно статистическое испытание, после чего произойдет останов. Такой останов выполняется путем введения в программу в конце статистического испытания проверки условия $\text{tg } 50 = 1$ (прямой угол состоит из 100 град, 50 град = 45°). Если это условие соблюдается, то предписывается выполнить останов вычислений. Соблюдение указанного условия будет иметь место в том случае, когда переключатель Р—ГРД—Г поставлен в среднее положение ГРД. Если же рассматриваемое условие не выполняется (а оно не соблюдается, когда переключатель Р—ГРД—Г поставлен в положение Р или Г, поскольку $\text{tg } 50 \text{ рад} \neq 1$ и $\text{tg } 50^\circ \neq 1$), то останов не происходит и осуществляется автоматический переход к выполнению следующего статистического испытания. Подпрограмма останова вычислений с помощью переключателя Р—ГРД—Г представлена ниже:

```

ПОДПРОГРАММА 8.5 — останов ГРД. Прекращение вычислений путем
установки переключателя Р—ГРД—Г в положение ГРД.
алг останов — ГРД
    арг ГРАДЫ
    рез останов
нач
    если  $\text{tg } 50 = 1$ 
        1 RX = TG (50) — 1
    кон
    то останов
        2 IF RX = 0 ELSE 5
            00. КНОП
            01. 5
            02. 0 (50)
            03. Ftg (tg 50)
            04. 1 (1, tg 50)
            05. — (tg 50 — 1)
        06. Fx = 0
        07. 09
    кон
    все
    ВОЗВРАТ
кон
ТЕСТ
РАДИАНЫ; НЖ В/О С/П; ВЫ —1.2718997; ГРАДЫ; НЖ В/О С/П;
ВЫ RX = 0 (3 с)

```

Перед остановом обычно всегда помещается в регистр Х и затем выдается значение количества выполненных испытаний, а также другие результаты. В практически используемых про-

граммах рассмотренная подпрограмма останова дополняется соответствующими командами, помещаемыми между командами 07.09 и 08.C/P (между БЕЙСИК-командами 2 и 3).

Упражнения

8.3.1. Составить программу определения количества неповторяющихся случайных чисел, генерируемых ДСЧ, использовав в этой программе останов с помощью переключателя Р—ГРД—Г.

8.3.2. Составить программу расчета гистограммы с 10 интервалами для вычисления коэффициента неравномерности распределения случайных чисел, выдаваемых ДСЧ. Предусмотреть в программе останов по заданному заранее количеству выданных случайных чисел и по усмотрению пользователя.

8.3.3. По составленной в предыдущем упражнении программе произвести испытания датчиков случайных чисел, характеристики которых приведены в табл. 8.1. Общее количество выдаваемых случайных чисел для каждого датчика выбрать равным 1000.

8.4. ОЦЕНКА ТОЧНОСТИ РЕЗУЛЬТАТОВ РЕШЕНИЯ ЗАДАЧИ ПО ПРОИЗВЕДЕННОМУ ЧИСЛУ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ

Пусть произведено N статистических испытаний и после обработки выходных данных получен некоторый результат \bar{z} (вероятность или среднее значение). Если обозначить искомый точный результат z , то отклонение полученного статистического результата от него составит $|\bar{z} - z| = \varepsilon$. Величину ε будем называть точностью полученного результата и понимать под ним максимальное отклонение полученного результата от его точного значения. Отклонение в принципе может быть сколь угодно большим, поэтому, говоря о максимальном отклонении, будем подразумевать, что вероятность его превышения достаточно мала. Обозначим β — вероятность того, что отклонение полученного результата \bar{z} от точного значения не превзойдет ε . Тогда можно записать

$$P(z - \varepsilon < \bar{z} < z + \varepsilon) = P(|\bar{z} - z| < \varepsilon) = \beta.$$

Вероятность β называется доверительной вероятностью, характеризующей надежность полученного решения с точностью ε .

Пусть искомый результат характеризуется математическим ожиданием m_z и дисперсией $D_z = \sigma_z^2$. В качестве оценки математического ожидания примем среднее арифметическое N произведенных статистических испытаний

$$\bar{z} = \frac{1}{N} \sum_{i=1}^N z_i.$$

Тогда согласно (8.6) дисперсия среднего значения

$$D[\bar{z}] = D\left[\frac{1}{N} \sum_{i=1}^N z_i\right] = \frac{1}{N^2} D\left[\sum_{i=1}^N z_i\right] = \frac{1}{N^2} ND_z = \frac{1}{N} \sigma_z^2.$$

При большом количестве испытаний N порядка нескольких десятков согласно центральной предельной теореме результат \bar{z}

можно считать распределенным по нормальному закону с математическим ожиданием m_z дисперсией σ_z^2/N . Для простоты обозначим: $m_z = m$ и $\sigma_z^2/N = D$. Тогда выражение для плотности вероятности \bar{z} можно записать

$$f(\bar{z}) = \frac{1}{\sqrt{2\pi D}} e^{-\frac{(\bar{z}-m)^2}{2D}}.$$

Вероятность попадания случайной величины \bar{z} в некоторый интервал $(m + \varepsilon, m - \varepsilon)$ может быть вычислена с помощью интеграла вероятностей — функции Лапласа:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2} dt;$$

$$\beta = P(m - \varepsilon < \bar{z} < m + \varepsilon) = \frac{1}{\sqrt{2\pi D}} \int_{m-\varepsilon}^{m+\varepsilon} e^{-\frac{(\bar{z}-m)^2}{2D}} d\bar{z}.$$

Произведя замену переменных:

$$t = (\bar{z} - m)/\sqrt{D}; \quad dt = d\bar{z}/\sqrt{D}; \quad d\bar{z} = \sqrt{D} dt; \quad \bar{z} - m = t\sqrt{D}; \\ t_1 = (m - \varepsilon - m)/\sqrt{D} = -\varepsilon/\sqrt{D}; \quad t_2 = (m + \varepsilon - m)/\sqrt{D} = \varepsilon/\sqrt{D},$$

получим

$$\beta = \frac{1}{\sqrt{2\pi}} \int_{-\varepsilon/\sqrt{D}}^{+\varepsilon/\sqrt{D}} e^{-t^2/2} dt = \frac{1}{\sqrt{2\pi}} \int_{-\varepsilon/\sqrt{D}}^0 e^{-t^2/2} dt + \\ + \frac{1}{\sqrt{2\pi}} \int_0^{+\varepsilon/\sqrt{D}} e^{-t^2/2} dt = \Phi(\varepsilon/\sqrt{D}) - \Phi(-\varepsilon/\sqrt{D}) = 2\Phi(\varepsilon/\sqrt{D}).$$

В результате придем к уравнению

$$\Phi(\varepsilon/\sqrt{D}) = \beta/2$$

или, используя обратную функцию Лапласа, получим

$$\varepsilon/\sqrt{D} = \Phi^{-1}(\beta/2).$$

Подставляя вместо D его значение σ_z^2/N , определим ε и окончательно получим

$$\varepsilon = \frac{\sigma_z \Phi^{-1}(\beta/2)}{\sqrt{N}}. \quad (8.12)$$

Значения обратной функции Лапласа приведены в табл. 8.3.

Таблица 8.3

Значения обратной функции Лапласа $\Phi^{-1}(\beta/2)$

β	0.900	0.950	0.960	0.970	0.980	0.990	0.995	0.997	0.998	0.999
$\Phi^{-1}\left(\frac{\beta}{2}\right)$	1.65	1.96	2.05	2.17	2.33	2.58	2.81	2.96	3.10	3.29

Выбирая из табл. 8.3 значения доверительной вероятности β , можно определить значение функции $\Phi^{-1}\left(\frac{\beta}{2}\right)$ и с помощью (8.12) получить оценку точности результата — среднего значения \bar{z} за N статистических испытаний.

Если результатом решения задачи является не среднее значение какой-либо величины, а вероятность появления некоторого события p , то аналогично можно получить выражение для вычисления ε и для этого случая. При одном статистическом испытании вероятность появления желаемого события может принимать только два значения: 0 и 1. В качестве оценки вероятности события при N испытаниях будем принимать частоту его появления или статистическую вероятность

$$\bar{p} = \frac{1}{N} \sum_{i=1}^N p_i; \quad D[\bar{p}] = \frac{1}{N^2} D\left[\sum_{i=1}^N p_i\right] = p(1-p)/N.$$

Полагая, что величина \bar{p} распределена по нормальному закону с математическим ожиданием p и дисперсией $p(1-p)/N$, находим вероятность ее попадания в интервал $(p - \varepsilon, p + \varepsilon)$:

$$\beta = P(p - \varepsilon < \bar{p} < p + \varepsilon) = 2\Phi(\beta/2)\left(\varepsilon / \sqrt{\frac{N}{p(1-p)}}\right),$$

откуда

$$\varepsilon = \sqrt{\frac{p(1-p)}{N}} \Phi^{-1}(\beta/2). \quad (8.13)$$

В отличие от предыдущего случая здесь необходимо знать статистическую вероятность (среднее значение частоты), а не дисперсию.

Итак, оценку точности полученных результатов в серии статистических испытаний можно получить с помощью формул (8.12) и (8.13), если к моменту окончания серии будут известны значения $z = \bar{z}$ или $p = \bar{p}$. Для оценки точности результатов статистических испытаний предлагаются следующие подпрограммы, реализующие формулы (8.12) и (8.13) на ПМК:

ПОДПРОГРАММА 8.6 — оценка среднего. Оценка точности среднего значения случайной величины z при N испытаниях.

Исходные данные:

 N — количество выполненных испытаний; $M[z]_N$ — среднее значение z за N испытаний; $M[z^2]_N$ — среднее значение z^2 за N испытаний; $\Phi^{-1}(\beta/2)$ — значение обратной функции Лапласа по табл. 8.3; β — надежность (доверительная вероятность) оценки точности результата $M[z]$.Результаты: ε — оценка точности результата (среднего значения); $m = M[z]_N \pm \varepsilon$.

Расчетные формулы:

$$D[z]_N = M[z^2]_N - M^2[z]_N = D_N = D;$$

$$\sigma_N = \sigma_N = \sqrt{D[z]_N};$$

$$\varepsilon = \sigma_N \Phi^{-1}(\beta/2) / \sqrt{N}.$$

алг оценка среднего

арг $R5 = N, R6 = M[z], R7 = M[z^2], RO = \Phi^{-1}(\beta/2)$
рез $RX = \varepsilon$

нач

$$D_N = M[z^2] - M^2[z]_N$$

1	$RX = R7 - R6 * * 2$
	00. $\pi X7(R7)$
	01. $\pi X6(R6, R7)$
	02. $Fx^2(R6 * * 2, R7)$
	03. $- (R7, R6 * * 2)$

$$\text{знат} = \sqrt{D_N/N} * \Phi^{-1}(\beta/2)$$

2	$RX = SQR(RX/R5) * RO$
	04. $\pi X5(R5, D)$
	05. $\div (D/R5)$
	06. $F \sqrt{(SQR(D/R5))}$
	07. $\pi X0(R0, SQR(D/R5))$
	08. $\times (\varepsilon)$

ВОЗВРАТ

3	RETURN
	09. B/O (ε)

кон

ТЕСТ

BB $R5 = N = 10000, R6 = M[z] = 10, R7 = M[z^2] = 101, RO = \Phi^{-1}(\beta/2) = 1.96$ НЖ B/O C/P; ВЫ $RX = \varepsilon = 1.96 - 02 (5 c)$ ПОДПРОГРАММА 8.7 — оценка вероятности. Оценка точности статистической вероятности p появления некоторого события за N испытаний.

Исходные данные:

 N — количество выполненных испытаний; p_N — статистическая вероятность появления события при N испытаниях; $\Phi^{-1}(\beta/2)$ — обратная функция Лапласа, согласно табл. 8.3.

Результаты:

 ε — оценка точности значения статистической вероятности при N испытаниях.

Расчетная формула:

$$\varepsilon = \sqrt{\frac{p(1-p)}{N}} \Phi^{-1}(\beta/2).$$

алг оценка вероятности
арг $RC = p$, $R4 = N$, $R0 = \Phi^{-1}(\beta/2)$
рез $RX = \epsilon$

нач

$\epsilon = \sqrt{p(1-p)/N} * \Phi^{-1}(\beta/2)$
 $1 RX = SQR(RC * (1 - RC) * R0)$
 00. 1 (1)
 01. пXC (RC, 1)
 02. -(1 - RC)
 03. пXC (R, 1 - RC)
 04. X (RC * (1 - RC))
 05. пX4 (R4, RC * (1 - RC))
 06. / (RC * (1 - RC)/R4)
 07. F Y' (SQR(06. RX))
 08. пX0 (R0, 07. RX)
 09. X (R0 * 07. RX)

ВОЗВРАТ

2 RETURN

10. В/О (ϵ)

кон

ТЕСТ

BB $RC = p = 0.9$, $R4 = N = 100$, $R0 = \Phi^{-1}(\beta/2) = 1.96$
 НЖ В/О С/П; ВЫ $RX = \epsilon = 5.88 -02$ (3 с)

Подпрограммы 8.6 и 8.7 размещаются в участке программной памяти ПМК, остающейся после размещения основной программы, осуществляющей проведение статистических испытаний. Если программу оценки точности не удается разместить в памяти ПМК, то она полностью или частично выполняется вручную после завершения серии испытаний.

8.5. ПРИМЕР ПОСТРОЕНИЯ НА ПМК СТАТИСТИЧЕСКОЙ МОДЕЛИ УСИЛИТЕЛЬНОГО КАСКАДА ДЛЯ ОЦЕНКИ ВЛИЯНИЯ РАЗБРОСА ЕГО ЭЛЕМЕНТОВ

8.5.1. ПОСТАНОВКА ЗАДАЧИ

Пусть разработана и отлажена на макете электронная схема — усилительный каскад на микросхеме — операционном усилителе (рис. 8.6). В результате разработки и отладки установлены номинальные значения сопротивления резисторов $R1 = 22 \cdot 10^3$ Ом и $R2 = 220 \cdot 10^3$ Ом, обеспечивающие требуемый коэффициент усиления каскада $K = 10 \pm 5\%$. Требуется оценить влияние разброса значений сопротивлений резисторов на коэффициент усиления каскада. Определить необходимость подбора резисторов в процессе серийного производства при регулировке.

Если учесть, что коэффициент усиления микросхемы — операционного

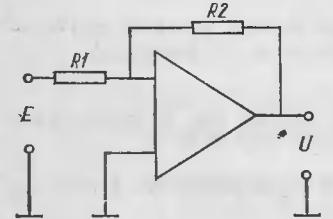


Рис. 8.6. Операционный усилитель на микросхеме: $K = U/E = R2/R1$ — коэффициент усиления каскада

усилителя очень большой ($>10^3$), то можно считать, что K зависит не от параметров самой микросхемы, а только от соотношения сопротивлений $R1$ и $R2$ и определяется выражением [7]

$$K = \frac{R2}{R1}. \quad (8.14)$$

Отсюда следует, что отклонения коэффициента усиления каскада будут зависеть только от отклонений сопротивлений резисторов $R1$ и $R2$ от их номиналов, которые имеют место при изготовлении. Резисторы массового применения выпускаются заводами-изготовителями с допусками ± 5 ; ± 10 ; $\pm 20\%$ от номинала, маркируемого на резисторе. Задачу учета влияния разброса сопротивлений резисторов на коэффициент усиления каскада нетрудно решить аналитически для наихудшего случая, когда отклонения резисторов $R1$ и $R2$ максимальны, а по знаку противоположны, что дает наибольшее изменение коэффициента усиления. Однако такие случаи практически никогда не встречаются и руководствоваться ими — это значит предъявлять неоправданно высокие требования к подбору элементов каскада. Поэтому наиболее целесообразным способом решения задачи является построение статистической модели усилительного каскада.

Датчик случайных величин будет выдавать значения сопротивлений резисторов $R1$ и $R2$, отклонения которых от номиналов моделируются датчиком при разыгрывании каждой очередной случайной величины. Для каждой пары значений $R1$ и $R2$, разыгранных датчиком, будет вычислен соответствующий коэффициент усиления каскада, а после этого — среднее значение коэффициента усиления и дисперсия за все произведенные статистические испытания. После проведения достаточного количества испытаний оценивается отклонение полученного среднего значения \bar{K}_N от истинного K и определяется возможный разброс этого отклонения σ_N .

8.5.2. СТРУКТУРНАЯ СХЕМА СТАТИСТИЧЕСКОЙ МОДЕЛИ

Структурная схема статистической модели усилительного каскада на микросхеме — операционном усилителе (рис. 8.7) состоит из трех групп блоков: датчика случайных значений сопротивлений резисторов R , блоков обработки результатов в каждом статистическом испытании, блоков обработки результатов серии статистических испытаний и оценки их точности. Рассмотрим их более подробно.

Датчик случайных значений сопротивлений резисторов представляет собой датчик случайных величин, распределение которых соответствует реальному закону распределения значений сопротивлений резисторов, выпускаемых заводами-изготовителями (блоки 1—4). Поскольку в процессе производства отклонение фактических значений сопротивлений резисторов от их номиналов

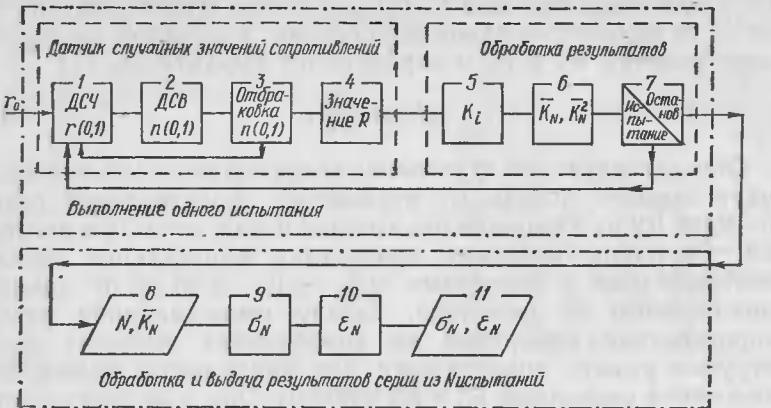


Рис. 8.7. Структурная схема статистической модели операционного усилителя: r_0 — начальное случайное число; k_i — коэффициент усиления для i -го испытания; N — число испытаний; $\bar{K}_N \pm \sigma$ — средний коэффициент усиления за N испытаний с точностью σ ; σ — среднее квадратическое отклонение K_i от \bar{K}_N

обусловлено очень большим числом причин, то указанные отклонения должны подчиняться нормальному закону) — предельному закону всех других законов распределения. При этом математическим ожиданием значения сопротивления будет его величина, обозначенная на резисторе. Вместе с ней здесь указывается и допуск в процентах — максимально допустимое отклонение от номинала, обозначенного на резисторе. При отклонениях, превышающих допуск, обозначенный на резисторе, последний бракуется. Поэтому отклонения, превышающие заданный допуск, не могут иметь места. Это означает, что распределение значений сопротивлений резисторов, выпускаемых заводом-изготовителем, подчиняется усеченному нормальному закону распределения. Характер усечения зависит как от допуска на отклонение (5; 10; 20 %), так и от технологии маркировки и отбраковки резисторов, используемых на заводе-изготовителе. В соответствии с изложенным датчик случайных значений сопротивлений резисторов должен состоять по крайней мере из четырех блоков:

1 — датчик случайных чисел, распределенных по равномерному закону в интервале $(0, 1)$ [на рис. 8.7 ДСЧ $z(0, 1)$];

2 — датчик нормированных нормально распределенных случайных величин в интервале $(-\infty, +\infty)$ с математическим ожиданием $m = 0$ и дисперсией $\sigma^2 = 1$ [на рис. 8.7 ДСВ $n(0, 1)$];

3 — блок отбраковки случайных величин, не подчиняющихся усеченномуциальному закону распределения;

4 — блок формирования значений сопротивлений резисторов, осуществляющего переход от нормированной случайной величины,

имитирующей отклонение сопротивления от номинала, к ненормированной, имитирующей значение сопротивления.

Для определения значений $R1$ и $R2$ обращение к датчику случайных чисел в каждом испытании производится дважды. При этом каждый раз должна быть произведена перестройка блоков 2, 3 и 4 на другие значения номиналов сопротивлений и допусков (допуски $R1$ и $R2$ могут быть различными). Подробно построение датчика случайных значений сопротивлений резисторов и его функционирование будет рассмотрено ниже.

Блоки обработки результатов в каждом статистическом испытании производят вычисление и обработку результатов в конце каждого из них, после чего вновь осуществляется обращение к датчику значений сопротивлений резисторов и выполняется следующее статистическое испытание. Эта группа состоит из трех блоков:

5 — блок вычисления коэффициента усиления каскада при каждом i -м испытании $K_i = R2_i/R1_i$;

6 — блок вычисления среднего значения коэффициента усиления за N испытаний \bar{K}_N и среднего значения квадрата коэффициента усиления \bar{K}_N^2 , необходимого в дальнейшем для вычисления дисперсии (этот блок разрабатывается на основе подпрограммы 8.4);

7 — блок останова серии испытаний по усмотрению пользователя (разрабатывается с использованием подпрограммы 8.5 — останов — ГРД).

Блоки обработки результатов серии статистических испытаний и оценки их точности составляют группу из четырех блоков:

8 — блок выдачи предварительных результатов серии статистических испытаний: N — количества выполненных испытаний, \bar{K}_N — среднего значения коэффициента усиления каскада (выдача указанных значений производится после каждого останова испытаний, произведенного по усмотрению пользователя);

9 — блок вычисления дисперсии D_N и среднего квадратического отклонения σ_N коэффициента усиления K_i от его среднего значения;

10 — блок оценки точности полученного значения среднего коэффициента усиления \bar{K}_N при N испытаниях ($K = \bar{K} \pm \sigma$) с помощью ранее рассмотренной подпрограммы 8.6 — оценка среднего;

11 — блок выдачи значений σ_N и σ .

Если точность определения среднего значения окажется недостаточной, пользователь принимает решение о продолжении серии статистических испытаний (нажимает клавишу С/П).

Основные блоки статистической модели были рассмотрены в предыдущих параграфах данной главы (подпрограммы 8.1, 8.2, 8.4, 8.5, 8.6). Новым является датчик случайных значений сопротивлений резисторов [за исключением датчика случайных чисел (блок 1) и датчика случайных величин (блок 2)].

В датчике случайных значений сопротивлений резисторов используются готовые подпрограммы: датчика случайных чисел, распределенных по равномерному закону в интервале $(0, 1)$ — подпрограмма 8.1 — ДСЧ и датчика нормированных нормально распределенных случайных величин — подпрограмма 8.2 — Гаусс (блоки 1 и 2). Рассмотрим построение и функционирование других блоков датчика случайных значений сопротивлений: блока отбраковки отклонений (блок 3) и блока формирования значений сопротивлений (блок 4). После этого разработаем соответствующую подпрограмму датчика сопротивлений.

Отбраковка нормированных случайных величин, распределенных по нормальному закону. Как уже отмечалось, отбраковка случайных величин, осуществляемая блоком 3, представляет собой операцию усечения нормального закона распределения. Характер усечения существенно зависит от технологии измерения и маркировки изготовленных резисторов. Рассмотрим две из возможных технологий:

технологию с фиксированным значением допуска (технология 1),

технологию с фиксированным значением номинала (технология 2).

Технология 1 (с фиксированным значением допуска) основана на использовании ряда номинальных значений сопротивлений (числовых коэффициентов, умноженных на любое число, кратное 10), установленного стандартами для каждого допуска. Элементы рядов выбираются таким образом, чтобы при отклонениях от номинала, превышающих среднее квадратическое σ , можно было бы изменить значение номинала и сделать его равным значению соседнего элемента ряда (справа или слева). При этом фактическое значение сопротивления окажется уже в пределах допуска для нового номинального значения сопротивления. Например, для 20 % допуска установлен ряд числовых коэффициентов: 1.0; 1.5; 2.2; 3.3; 4.7; 6.8. Если отклонение значения $R = 2.2 \text{ к}\Omega$ превзойдет допуск в 20 %, т. е. $R = 2.2 - 0.44 = 1.76 \text{ к}\Omega$, то можно изменить его номинал на 1.5 кОм. Тогда отклонение от номинала составит $(1.76 - 1.5)/1.5 = 0.14 = 14\%$. Если же значение сопротивления будет увеличено в большую сторону, т. е. $R = 2.2 + 0.44 = 2.64 \text{ к}\Omega$, то можно изменить номинал на 3.3 кОм. Отклонение при этом $(3.3 - 2.64)/3.3 = 0.087 = 8.7\%$. При такой технологии отбраковки и маркировки отклонения от номинала будут подчиняться нормальному закону распределения, усеченному на уровне $\pm\sigma$. При этом значения допуска d , выраженные в сотых долях процента ($0.2, 0.1, 0.05$), будут соответствовать на оси абсцисс кривой плотности распределения в одной и той же точке, расположенной на расстоянии σ от начала координат (рис. 8.8).

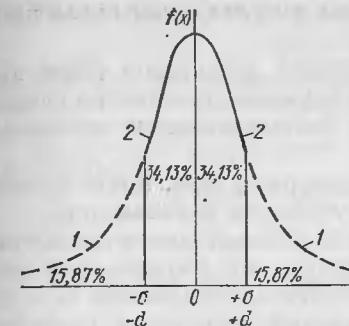


Рис. 8.8. Распределение отклонений сопротивлений резисторов от номиналов. Технология 1 с фиксированным значением допуска:
1 — область отклонений с переходом в соседний номинал; 2 — область допустимых отклонений от заданного номинала

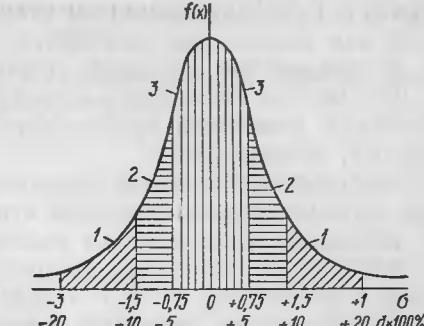


Рис. 8.9. Распределение отклонений сопротивлений от номиналов. Технология 2 с фиксированным значением номинала:
1 — область отклонений с 20 % допуском; 2 — область отклонений с 10 % допуском; 3 — область отклонений с 5 % допуском

Рассмотренная технология обеспечивает выход 68,26 % резисторов, соответствующих первоначально заданному номиналу, а остальные — соответствующие соседним номиналам (лишь менее 0,3 % будут отбраковываться с отклонениями, превосходящими допуски соседних номиналов).

Технология 2 (с фиксированным значением номинала) устанавливает фиксированное значение номинала с максимальным допуском в 20 %, соответствующим отклонению $\pm 3\sigma$. Изготавливаемые резисторы установленного номинала сортируются по фактическим отклонениям, укладывающимся в 5; 10; 20 % допуска. При этом отбраковывается 0,3 % резисторов, номиналы которых не укладываются в 20 %-ный допуск.

Кривая плотности распределения усеченного нормального закона в случае использования технологии 2 приведена на рис. 8.9.

Для решения вопроса о том, по какой технологии изготовлены и промаркованы резисторы, достаточно измерить фактические значения сопротивлений партии резисторов для каждого допуска одного номинала и рассчитать гистограмму распределения отклонений, используя программу 6.2. Измерение значений сопротивлений партии в 100—300 резисторов все же намного проще, чем их перепайка на макете и проведение натурных испытаний, тем более, что исследовать закон распределения исходных случайных значений нужно только один раз (при построении статистической модели). В зависимости от технологии маркировки резисторов в статистическую модель будет включен тот или иной блок отбраковки случайных величин n ($0, 1$). В качестве примера используем в статистической модели блок отбраковки, соответствующий тех-

нологии 1 (с фиксированным значением допуска), как более простой для реализации на ПМК.

Формирование значений сопротивлений резисторов (блок 4).

Формирование каждого очередного случайного значения сопротивления резистора производится с использованием исходных данных, включающих:

номинальное значение сопротивления резистора, принимаемое как математическое ожидание его случайного значения m ;

допуск на отклонение от номинала d в сотых долях процента;

случайную величину, распределенную по нормальному закону с дисперсией $\sigma^2 = 1$ и математическим ожиданием $m = 0$.

Формирование значений сопротивлений резисторов сводится к преобразованию случайных величин, распределенных по нормированномуциальному закону в ненормированные случайные величины. Для такого преобразования необходимо знать математическое ожидание ненормированной величины (оно нам известно — это номинал сопротивления) и ее дисперсию σ^2 . Последняя нам неизвестна, ее необходимо определить, используя известное значение допуска d .

Максимальное отклонение значения сопротивления от номинала m при допуске d равно md . В соответствии с технологией 1 (рис. 8.8) оно соответствует отклонению σ на абсциссе кривой плотности распределения: $\sigma = md$. Используя формулу преобразования (8.11а) нормированной нормально распределенной случайной величины $n(0, 1)$ в ненормированную:

$$n(m, \sigma) = m + \sigma n(0, 1)$$

и подставляя в нее $\sigma = md$, получим выражение для случайного значения сопротивления резистора с номиналом m и допуском d

$$n(m, d) = m (1 + dn(0, 1)),$$

на основе которого можно будет построить блок формирования случайного значения сопротивления (блок 4).

С учетом изложенного подпрограмма датчика случайных значений сопротивлений резисторов (в принципе она будет пригодна и для других пассивных элементов электронных схем: конденсаторов, индуктивностей и др.) будет выглядеть следующим образом:

ПОДПРОГРАММА 8.8 — значение R. Датчик случайных значений сопротивлений резисторов, распределенных по усеченному нормальному закону (технология 1 с фиксированным допуском).

Исходные данные:

r_0 — начальное значение случайного числа, распределенного по равномерному закону в интервале $(0, 1)$;

m — номинальное значение сопротивления резистора, равное его математическому ожиданию;

d — допуск на отклонение в сотых долях процента.

Результаты:

$n(m, d)$ — случайное значение сопротивления резистора с номиналом m и допуском d .

Расчетная формула:

$$n(m, d) = m (1 + dn(0, 1)),$$

где $n(0, 1)$ — нормированная нормально распределенная случайная величина с $m = 0$ и $\sigma = 1$, удовлетворяющая условию $n(0, 1) < \sigma$.

Условия отбраковки: $n(0, 1) \geq \sigma = 1$.

Используемая подпрограмма: 8.2 — Гаусс.

```

алг значение R
    арг RA = m, RB = d, RD = r0
    рез RX = n(m, d)
нач RD = r0 = r, R0 = n(0, 1)
иц
    Гаусс (арг RD = r, рез RX = n(0, 1))
        1 SUB
        00. пXD 06. K {x} 12. — 18. ÷ 24. 1
        01. 1 07. хΠD 13. × 19. × 25. —
        02. 1 08. 4 14. Fln 20. F √ 26. КЗН
        03. × 09. × 15. Fπ 21. пXD 27. ×
        04. Fπ 10. 1 16. /—/ 22. 2
        05. + 11. пXD 17. 2 23. ×
    n(0, 1) = знач
    2 R0 = RX
    28. xΠ0 (R0 = n(0, 1))

иц до |n(0, 1)| < 1
    3 RX = ABS (RX) — 1
        29. K |x| (ABS (n(0, 1)))
        30. 1 (1, ABS (n(0, 1)))
        31. — (ABS (n(0, 1)) — 1)
    4 IF RX = 0 ELSE 1
        32. Fx < 0
        33. 00
    n(m, d) = m (1 + dn(0, 1))
    5 RX = RA * (1 + RB * R0)
        34. пXB (RB)
        35. пX0 (R0, RB)
        36. × (RB * R0)
        37. 1 (1, RB * R0)
        38. + (1 + RB * R0)
        39. пXA (RA, 1 + RB * R0)
        40. × (RA * (1 + RB * R0))

    ВОЗВРАТ
    6 RETURN
    41. B/O (n(m, d))

кон
TEST
1) ВВ RA = m = 110000, RB = d = 0.2, RD = r0 = 0.5
   НЖ В/О С/П; ВЫ RX = n(m, d) = 117971.97 (15 c)
   НЖ В/О С/П; ВЫ 91511.101
   НЖ В/О С/П; ВЫ 100445.46
   НЖ В/О С/П; ВЫ 127773.22
2) ВВ RB = 0.1, RD = 0.5
   НЖ В/О С/П; ВЫ 113985.99
   НЖ В/О С/П; ВЫ 100755.56
   НЖ В/О С/П; ВЫ 105222.72
3) ВВ RB = d = 0.05; RD = 0.5; НЖ В/О С/П; ВЫ 111992.99

```

8.5.4. СТАТИСТИЧЕСКАЯ МОДЕЛЬ КАСКАДА — ОПЕРАЦИОННОГО УСИЛИТЕЛЯ НА МИКРОСХЕМЕ

Выше были рассмотрены все подпрограммы и вспомогательные программы, необходимые для построения статистической модели каскада — операционного усилителя на микросхеме. Общая программа этой модели будет выглядеть следующим образом:

ПРОГРАММА 8.9 — каскад. Статистическая модель каскада на микросхеме — операционном усилителе.

Исходные данные:

r_0 — начальное значение датчика случайных чисел;

m_1 и m_2 — номиналы резисторов согласно маркировке;

d_1 и d_2 — их допуски согласно маркировке в сотых долях процента;

$\Phi^{-1}(\beta/2)$ — значение обратной функции Лапласа согласно табл. 8.3;

β — задаваемая надежность оценки точности результата решения — среднего значения коэффициента усиления каскада \bar{K}_N за N испытаний.

Результаты:

N — число произведенных статистических испытаний;

\bar{K}_N — средний коэффициент усиления каскада за N испытаний;

\bar{K}_N^2 — среднее значение квадрата коэффициента усиления за N испытаний;

σ_N — среднее квадратическое отклонение коэффициента усиления K_i от среднего значения \bar{K}_N ;

ϵ — точность полученного значения \bar{K}_N ($K = \bar{K}_N \pm \epsilon$).

Используемые подпрограммы: 8.4 — среднее, 8.6 — оценка среднего, 8.8 — значение R .

алг каскад

арг $R1 = m_1$, $R2 = m_2$, $R3 = d_1$, $R4 = d_2$, $RD = r_0$, $RE = \Phi^{-1}(\beta/2)$

рез $RX = N$, $R6 = \bar{K}_N$, $R7 = \bar{K}_N^2$, $RC = \sigma_N$

нач $RO = n(0, 1)$, $R8 = n(m_1, d_1)$, $R8 = n(m_2, d_2)$, $R9 = (N - 1)/N$, $RA = m_1$,

$RA = m_2$, $RB = d_1$, $RB = d_2$, $R5 = N$

ВВ $R1 = m_1$, $R2 = m_2$, $R3 = d_1$, $R4 = d_2$, $RD = r_0$, $RE = \Phi^{-1}(\beta/2)$

РАДИАНЫ; НЖ В/О С/П

$N = 0$

1 $R5 = 0$

00. $Cx(0)$
01. $x\Pi 5(R5 = 0)$

$\bar{K}_N = 0$

2 $R6 = 0$

02. $x\Pi 6(R6 = 0)$

$\bar{K}_N^2 = 0$

3 $R7 = 0$

03. $x\Pi 7(R7 = 0)$

нц

значение R (арг $RD = r$, $RA = m_1$, $RB = d_1$ рез $RX = n(m, d)$, пр RO)
4 $RA = R1$

04. $\Pi X1(R1)$
05. $x\Pi A(R1 = RA)$

5 $RB = R3$

06. $\Pi X3(R3)$
07. $x\Pi B(R3 = RB)$

6 GOSUB 16

08. ПП
09. 54 ($n(m, d)$)

$n(m, d) = \text{знач}$

7 $R8 = RX$

10. $x\Pi 8(R8 = n(m_1, d_1))$

значение R (арг $RD = r$, $RA = m_2$, $RB = d_2$, рез $RX = n(m, d)$, пр RO)

8 $RA = R2$

11. $\Pi X2(R2)$
12. $x\Pi A(R2 = m_2)$

9 $RB = R4$

13. $\Pi X4(R4)$
14. $x\Pi B(RB = d_2)$

10 GOSUB 16

15. ПП
16. 54 ($n(m_2, d_2)$)

$K = n(m_2, d_2)/n(m_1, d_1)$

11 $Rx = RX/R8$

17. $\Pi X8(R8, n(m_2, d_2))$
18. $\div(n(m_2, d_2)/R8)$

среднее (арг $RX = K$, $R5 = N - 1$, $R6 = \bar{K}_{N-1}$, $R7 = \bar{K}_{N-1}^2$, рез $R5 = N$, $R6 = \bar{K}_N$, $R7 = \bar{K}_N^2$, пр $R8, R9$)

12 SUB

19. $x\Pi 8$	24. $x\Pi 5$	29. $\Pi X8$	34. $\Pi X9$	39. $\Pi X5$
20. $\Pi X5$	25. \div	30. $\Pi X5$	35. $\Pi X7$	40. \div
21. $\Pi X5$	26. $x\Pi 9$	31. \div	36. \times	41. $+$
22. 1	27. $\Pi X6$	32. $+$	37. $\Pi X8$	42. $x\Pi 7$
23. +	28. \times	33. $x\Pi 6$	38. Fx^2	

кц

до ГРАДЫ

останов — ГРД (арг ГРАДЫ, рез останов)

13 SUB

43. 5	45. Ftg	47. —	49. 52
44. 0	46. 1	48. $Fx = 0$	

ВЫ $RX = N$

14 STOP

50. $\Pi X5(R5)$
51. С/П (N)

если N недостаточно

то * продолжение вычислений * РАДИАНЫ; НЖ С/П

15 GOTO 4

52. БП
53. 04

все

16 SUB	значение R			
54. ΠXD	63. \times	72. \div	81. \times	90. \times
55. 1	64. 1	73. \times	82. $x\Pi O$	91. 1
56. 1	65. ΠXD	74. FV	83. $K x $	92. +
57. \times	66. —	75. ΠXD	84. 1	93. ΠXA
58. $F\pi$	67. \times	76. 2	85. —	94. \times
59. +	68. Fln	77. \times	86. $Fx < 0$	95. В/О
60. $K \{x\}$	69. $F\pi$	78. 1	87. 54	
61. $x\Pi D$	70. $/-$	79. —	88. ΠXB	
62. 4	71. 2	80. $K3H$	89. $\Pi X0$	

НЖ БП 96 С/П

оценка среднего (арг $R5 = N$, $R6 = \bar{K}_N$, $R7 = \bar{K}_N^2$, рез $RC = 0$)

17 SUB

96. $\Pi X7$	98. Fx^2	0.0. $F V$
97. $\Pi X6$	99. —	1.1. $x\Pi C$

ВЫ $RX = \sigma$

18 STOP

2.2. С/П (σ)

* Вычисление ε вручную * НЖ пX5 (N, σ)
 $FV(\sqrt{N}, \sigma)$
 $\div (\sigma/\sqrt{N})$
 $\text{пХЕ}(\Phi^{-1}, \sigma/\sqrt{N})$
 $\times (\varepsilon = \sigma/\sqrt{N} * \Phi^{-1})$

ВЫ RX = ε , R6 = \bar{K}_N
 если точность ε недостаточна
 то РАДИАНЫ; НЖ С/П

19 GOTO 4

3.3. БП
 4.4. 04

все

кон
ИНСТРУКЦИЯBB R1 = m_1 , R2 = m_2 , R3 = d_1 , R4 = d_2 , RD = r_0 , RE = $\Phi^{-1}(\beta/2)$

РАДИАНЫ; НЖ В/О С/П

если останов
то ГРАДЫВЫ RX = N если N недостаточното * продолжение вычислений *
 РАДИАНЫ; НЖ С/Пиначе НЖ БП 96 С/П; ВЫ RX = σ
 * вычисление ε вручную * НЖ пX5 (N, σ)
 $FV(\sqrt{N}, \sigma)$
 $\div (\sigma/\sqrt{N})$
 $\text{пХЕ}(\Phi^{-1}, \sigma/\sqrt{N})$
 $\times (\varepsilon = (\sigma/\sqrt{N}) \Phi^{-1}(\beta/2))$
ВЫ RX = ε R6 = \bar{K} если точность ε недостаточнато * продолжение вычислений *
 РАДИАНЫ; НЖ С/П

все

TEST
ГРАДЫ
 BB RD = $r_0 = 0.5$, R1 = $m_1 = 22000$, R2 = $m_2 = 220000$, R3 = $d_1 = 0.05$,
 $R4 = d_2 = 0.2$, RE = $\Phi^{-1}(\beta/2) = 1.96$

НЖ В/О С/П; ВЫ N = 1 (45 с), R6 = $\bar{K}_1 = 8.1711453$, R7 = $\bar{K}_1^2 = 66.767616$ НЖ С/П; ВЫ RX = N = 2, R6 = $\bar{K}_1 = 10.022363$, R7 = $\bar{K}_1^2 = 103.87477$ НЖ БП 96 С/П; ВЫ RX = $\sigma = 1.8512185$;НЖ пX5 FV ÷ пХЕ ×; ВЫ $\varepsilon = 2.565658$

П р и м е ч а н и е. Подпрограмма 8.6 — оценка среднего не умещается в памяти ПМК полностью. Поэтому с помощью указанной подпрограммы вычисляется только значение σ , а вычисление ε производится после этого вручную по формуле $\varepsilon = (\sigma/\sqrt{N}) * \Phi^{-1}(\beta/2)$.

8.5.5. ПРОВЕДЕНИЕ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ НА МОДЕЛИ УСИЛИТЕЛЬНОГО КАСКАДА

Пусть на макете усилительного каскада при точно подобранных сопротивлениях резисторов $R1 = 22 \cdot 10^3$ Ом и $R2 = 220 \times$

Результаты испытаний статистической модели усилительного каскада
(коэффициент усиления $K = R2/R1$; $\beta = 0.95$, $\Phi^{-1}(\beta/2) = 1.96$)

Исходные данные		Результаты			
$R2 \cdot 10^3$, Ом	$R1 \cdot 10^3$, Ом	Число испытаний N	Средний коэффициент усиления $\bar{K} \pm \varepsilon$	Среднее квадратическое отклонение σ_N	Время решения, ч
$220 \pm 5\%$	$22 \pm 5\%$	1273	10.006 ± 0.021	0.376	18
	$22 \pm 10\%$	892	10.037 ± 0.039	0.587	15.5
	$22 \pm 20\%$	1035	10.122 ± 0.070	1.098	16.5
	$22 \pm 5\%$	1161	10.031 ± 0.034	0.586	17.5
$220 \pm 10\%$	$22 \pm 10\%$	1053	10.023 ± 0.045	0.745	16.5
	$22 \pm 20\%$	1202	10.122 ± 0.068	1.195	18
	$22 \pm 5\%$	1003	9.997 ± 0.069	1.109	16
	$22 \pm 10\%$	938	10.035 ± 0.077	1.197	16
$220 \pm 10\%$	$22 \pm 20\%$	1077	10.111 ± 0.090	1.511	16.5

$\times 10^3$ Ом получился коэффициент усиления $K = 10$. В серийном производстве с учетом разброса сопротивлений $R1$ и $R2$ коэффициент усиления должен не выходить из пределов $K = 10 \pm 5\%$. Если применить резисторы массового производства с наименьшим допуском в 5 %, то максимальное отклонение K от 10 будет иметь место в двух случаях:

— когда $R1 = 22 - 1.1 = 20.9$ кОм, а $R2 = 220 + 11 = 231$ кОм, то $K = R2/R1 = 231/20.9 = 11.05$;

— когда $R1 = 22 + 1.1 = 23.1$ кОм, а $R2 = 220 - 11 = 209$ кОм, то $K = 209/23.1 = 9.05$, т. е. в самом наихудшем случае отклонение коэффициента усиления K от заданного значения будет превышать 11 % (вместо допустимых 5 %). Отсюда напрашивается вывод, что применять в усилительном каскаде резисторы массового производства с наименьшим допуском в 5 % нельзя. Необходим их специальный подбор при настройке и регулировке усилителя.

Однако остается неясным, насколько часто при использовании резисторов с 5 %-ным допуском коэффициент усиления каскада будет выходить за допустимые пределы. Ответ на этот вопрос можно получить путем проведения статистических испытаний на разработанной статистической модели каскада и реализованной на ПМК в виде программы 8.9 — каскад. Результаты испытаний на модели с различными значениями допусков сопротивлений резисторов $R1$ и $R2$ приведены в табл. 8.4. В отличие от всех предшествующих таблиц с результатами вычислений, в данной таблице значения величин даны с точностью до трех знаков после десятичной точки. Это, с одной стороны, вызвано тем, что повторить в точности полученные результаты в данном случае затруднительно, поскольку нельзя угадать, какое точное количество

испытаний уже выполнено перед остановом с помощью переключателя Р—ГРД—Г. С другой стороны, точность полученных результатов, характеризуемая величиной ε , намного ниже отбрасываемых при округлении знаков.

Результаты испытаний показывают следующее:

1. При количестве испытаний порядка 1000 среднее значение коэффициента усиления незначительно отличается от заданного (0.06 % для резисторов с допуском 5 %) при достаточной точности вычислений (0.2 % для резисторов с допуском 5 %).

2. Среднее квадратическое отклонение коэффициента усиления K от среднего при использовании резисторов R_1 и R_2 с допуском 5 % составляет 0.376. Это значит, что почти в 70 % случаев коэффициент усиления будет находиться в пределах $K = 9.63 \div 10.38$ (т. е. его отклонения не превзойдут 3.86 %). Отсюда следует, что в процессе серийного производства можно использовать резисторы с допуском 5 % в качестве элементов усилительного каскада с проверкой коэффициента усиления последнего на каждом экземпляре усилителя. При этом в большинстве случаев подбор сопротивлений не потребуется. В отдельных случаях достаточно будет подобрать один резистор R_1 , влияние которого на изменение K оказывается несколько больше, чем влияние R_2 .

8.6. ПРИМЕР ПОСТРОЕНИЯ НА ПМК СТАТИСТИЧЕСКОЙ МОДЕЛИ КАНАЛА СВЯЗИ

Пусть на вход одноканального направления связи поступает поток заявок на передачу телеграфных сообщений с интенсивностью λ заявок в час. Плотность вероятности интервалов времени между моментами поступления заявок подчиняется экспоненциальному закону распределения

$$f(t) = \lambda e^{-\lambda t}.$$

Передача сообщений (обслуживание заявок) выполняется с интенсивностью μ сообщений в час (интенсивность обслуживания). Плотность вероятности интервалов времени передачи τ (от начала до конца передачи сообщения) также подчиняется экспоненциальному закону распределения

$$f(t) = \mu e^{-\mu t}.$$

Поступающая заявка либо сразу обрабатывается, либо становится в очередь. При нахождении в очереди больше допустимого времени ($t_{ож} > T_d$) заявка покидает очередь не обслуженной.

Требуется разработать алгоритм и программу для ПМК статистической модели одноканального направления связи, позволяющую определить:

статистическую вероятность обслуживания заявки p с оценкой ее точности ε , при времени ожидания в очереди $t_{ож}$ меньше допустимого T_d ;

среднее время ожидания заявки в очереди $T_{ож}$;

коэффициент простого канала связи за время поступления заявок K_n .

8.6.1. СТРУКТУРНАЯ СХЕМА СТАТИСТИЧЕСКОЙ МОДЕЛИ КАНАЛА СВЯЗИ

Структурная схема статистической модели канала связи (рис. 8.10) состоит из трех групп блоков, выполняющих функции: датчика случайного времени поступления и окончания обработки заявок; обработки результатов в конце каждого статистического испытания; обработки и выдачи результатов серии статистических испытаний.

Датчик случайного времени (блоки 1—4) генерирует значения моментов времени поступления заявок T и времени окончания их обслуживания T_k . Интервалы времени между моментами поступления заявок t и моментами окончания их обслуживания τ генерируются с помощью датчика случайных величин, распределенных по экспоненциальному закону (блоки 1—2), а их преобразование в соответствующие моменты времени поступления и окончания обслуживания заявок (блоки 3—4). Разыгрывание случайных величин $t = e(\lambda)$, имитирующих интервал между поступлениями заявок, и $\tau = e(\mu)$, имитирующих интервал между началом и концом обслуживания заявки, производится одним и тем же датчиком случайных величин. Поэтому в каждом статистическом

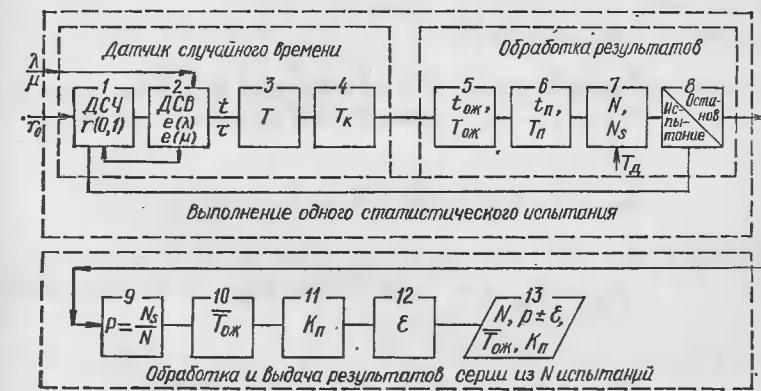


Рис. 8.10. Структурная схема статистической модели одноканального направления связи:

t_0 — начальное случайное число; λ (μ) — интенсивность потока заявок (обслуживания), заявок/час; t (τ) — интервал времени между моментами поступления заявок (началом и концом их обслуживания); T (T_k) — момент поступления очередной заявки; $T_{ож}$ (T_p) — суммарное время ожидания простоя в течение времени поступления N заявок; P — вероятность обслуживания заявки; ε — точность ее определения; $T_{ож}$ — среднее время ожидания заявки в очереди; K_n — коэффициент простой системы

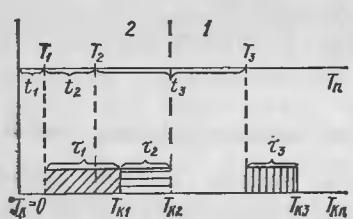


Рис. 8.11. Временная диаграмма работы канала связи:
 t_1, t_2, t_3 — интервалы времени между поступлениями заявок;
 τ_1, τ_2, τ_3 — интервалы времени, необходимые для их обслуживания; T_1, T_2, T_3 — моменты времени поступления заявок; T_{k1}, T_{k2}, T_{k3} — моменты времени окончания их обслуживания; I — случай 1: $T_3 \geq T_{k2}$; $T_{k3} = T_3 + \tau_3$; 2 — случай 2: $T_2 < T_{k1}$; $T_{k2} = T_{k1} + \tau_2$

или $T_i = T_{i-1} + t_i$, выдаваемые блоком 3.

3. Заявки обслуживаются в интервалы времени $\tau_1, \tau_2, \dots, \tau_i, \dots, \tau_N$, имитируемые датчиком случайных величин (блоки 1—2).

4. Обслуживание заявок заканчивается в моменты времени $T_{k1}, T_{k2}, \dots, T_{ki}, T_{k2}$, определяемые в блоке 4 по-разному для двух случаев;

случай 1 (см. рис. 8.11):

$$T_{k1} = T_1 + \tau_1 \text{ при } T_1 \geq T_{k0} (T_{k0} = 0);$$

$$T_{k3} = T_3 + \tau_3 \text{ при } T_3 \geq T_{k2};$$

$$\dots$$

$$T_{ki} = T_i + \tau_i \text{ при } T_i \geq T_{k(i-1)};$$

случай 2

$$T_{k2} = T_{k1} + \tau_2 \text{ при } T_2 < T_{k1};$$

$$\dots$$

$$T_{ki} = T_{k(i-1)} + \tau_i \text{ при } T_i < T_{k(i-1)}$$

или, объединяя оба случая, можно записать

$$T_k = \begin{cases} T_i + \tau_i & \text{при } T_i \geq T_{k(i-1)}; \\ T_{k(i-1)} + \tau_i & \text{при } T_i < T_{k(i-1)}. \end{cases}$$

Обработка результатов в конце каждого статистического испытания осуществляется блоками 5—8. После каждого статистического испытания вновь производится обращение к датчику случай-

испытаний обращение к датчику случайных величин производится дважды.

Функционирование датчика случайного времени (временная диаграмма — рис. 8.11) производится следующим образом:

1. Сообщения поступают через интервалы времени t_1, t_2, \dots, t_n , имитируемые датчиком случайных величин (блоки 1—2).

2. Заявки поступают в моменты времени:

$$\begin{aligned} T_1 &= t_1; \\ T_2 &= t_1 + t_2; \\ &\dots \\ T_i &= t_1 + t_2 + \dots + t_i = \sum_{i=1}^i t_i, \end{aligned}$$

ного времени и начинает выполняться новое, очередное статистическое испытание. При этом обработка результатов состоит из четырех этапов.

1. Определяется время ожидания заявки в очереди (блок 5):

$$t_{\text{ож2}} = T_{k1} - T_{k2} \text{ при } T_2 < T_{k1}; \quad t_{\text{ож3}} = 0 \text{ при } T_3 \geq T_{k2}$$

или в общем виде:

$$t_{\text{ож}} = \begin{cases} T_{k(i-1)} - T_i & \text{при } T_2 < T_{k1} \text{ и } t_{\text{ож}} < T_d; \\ 0 & \text{при } T_2 \geq T_{k1} \text{ и } t_{\text{ож}} \geq T_d, \end{cases}$$

где T_d — допустимое время ожидания в очереди.

В этом же блоке определяется суммарное время ожидания $T_{\text{ож}}$ для всех N поступивших заявок:

$$T_{\text{ож}} = \sum_{i=1}^N t_{\text{ож}i}.$$

2. Определяется время простоя канала связи t_p между моментами окончания обслуживания предыдущей заявки и поступления очередной (блок 6):

$$t_{\text{пп3}} = T_3 - T_{k2} \text{ при } T_3 \geq T_{k2}; \quad t_{\text{пп2}} = 0 \text{ при } T_2 < T_{k1}$$

или в общем виде:

$$t_{\text{пп}i} = \begin{cases} T_i - T_{k(i-1)}, & T_i \geq T_{k(i-1)}; \\ 0, & T_i < T_{k(i-1)}. \end{cases}$$

В этом же блоке определяется и суммарное время простоя за все время поступления N заявок:

$$T_p = \sum_{i=1}^N t_{\text{пп}i}.$$

3. Подсчитывается число поступивших N и обслуженных N_s заявок, исходя из соотношений:

$$N = N + 1 \text{ (во всех случаях);}$$

$$N_s = N_s + 1 \text{ при } t_{\text{ож}} < T_d; \quad N_s = 0 \text{ при } t_{\text{ож}} \geq T_d.$$

4. По окончании каждого статистического испытания предусматривается либо их прекращение по усмотрению пользователя (путем перевода переключателя Р—ГРД—Г в положение ГРД), либо переход к выполнению очередного статистического испытания. В случае прекращения статистических испытаний по усмотрению пользователя осуществляется автоматических переход к обработке и выдаче результатов серии произведенных N испытаний.

Обработка и выдача результатов серии статистических испытаний осуществляется в блоках 9—13, предусматривающих вычисление и выдачу выходных данных, получаемых с помощью статистической модели одноканального направления связи.

Такими данными являются:

1. Статистическая вероятность обслуживания заявки (блок 9)

$$p = N_s/N.$$

2. Среднее время ожидания одной выполненной заявки (блок 10)

$$\bar{T}_{\text{ож}} = T_{\text{ож}}/N.$$

3. Коэффициент простоя канала связи (блок 11)

$$K_{\text{п}} = T_{\text{п}}/T,$$

где T — общее время функционирования системы в течение поступления N заявок.

4. Оценка точности полученного значения вероятности

$$\epsilon = \sqrt{\frac{p(1-p)}{N}} \Phi^{-1}\left(\frac{\beta}{2}\right),$$

где $\Phi^{-1}(\beta/2)$ — обратная функция Лапласа (согласно табл. 8.3); β — задаваемая надежность вычисления ϵ .

8.6.2. РАЗРАБОТКА ПРОГРАММЫ СТАТИСТИЧЕСКОЙ МОДЕЛИ ОДНОКАНАЛЬНОГО НАПРАВЛЕНИЯ СВЯЗИ

С учетом изложенного статистическая модель одноканального направления связи будет реализована на ПМК с помощью следующей программы:

ПРОГРАММА 8.10 — канал. Статистическая модель одноканального направления связи.

Исходные данные:

r_0 — начальное значение случайного числа ДСЧ;

λ — интенсивность потока заявок на передачу сообщений, заяв./ч;

μ — интенсивность обслуживания заявок передачи сообщений, заяв./ч;

$\Phi^{-1}(\beta/2)$ — обратная функция Лапласа (согласно табл. 8.3),

где β — задаваемая надежность оценки точности ϵ определения вероятности обслуживания заявки.

Результаты:

N — число произведенных статистических испытаний;

p — статистическая вероятность обслуживания заявок при $t_{\text{ож}} < T_d$, где

T_d — допустимое время ожидания заявки в очереди;

$T_{\text{ож}}$ — среднее время ожидания заявок в очереди;

$K_{\text{п}}$ — коэффициент простоя канала связи;

ϵ — оценка точности результата p ($p_t = p \pm \epsilon$).

Расчетные формулы:

1. $T = T + t$, где T — момент поступления очередной заявки; t — интервал времени между моментами поступления предыдущей и последующей заявок.

2. $N = N + 1$ — счетчик числа поступивших заявок.

$$T_k = \begin{cases} T + \tau & \text{при } T \geq T_k; \\ T_k + \tau & \text{при } T < T_k, \end{cases}$$

где T_k — время окончания обслуживания предыдущей заявки; τ — время обслуживания текущей заявки.

4. $t_{\text{п}} = T - T_k$ при $T \geq T_k$, где $t_{\text{п}}$ — время простоя канала связи между окончанием передачи предыдущего сообщения и поступлением заявки на передачу очередного.

5. $T_{\text{п}} = T_{\text{п}} + t_{\text{п}}$, где $T_{\text{п}}$ — суммарное время простоя в течение времени проведения N испытаний.

6.

$$t_{\text{ож}} = \begin{cases} T_k - T & \text{при } T < T_k \text{ и } t_{\text{ож}} < T_d; \\ 0 & \text{при } t_{\text{ож}} \geq T_d, \end{cases}$$

где $t_{\text{ож}}$ — время ожидания заявки в очереди, T_d — допустимое время ожидания.

7. $T_{\text{ож}} = T_{\text{ож}} + t_{\text{ож}}$, где $T_{\text{ож}}$ — суммарное время ожидания за время поступления N заявок.

8.

$$N_s = \begin{cases} N_s + 1 & \text{при } T \geq T_k; \\ N_s + 1 & \text{при } T < T_k \text{ и } t_{\text{ож}} < T_d; \\ 0 & \text{при } t_{\text{ож}} \geq T_d, \end{cases}$$

где N_s — число обслуженных заявок из N поступивших.

9. $p = N_s/N.$

10.

$$\bar{T}_{\text{ож}} = T_{\text{ож}}/N,$$

где $\bar{T}_{\text{ож}}$ — среднее время ожидания заявок в очереди.

11.

$$K_{\text{п}} = T_{\text{п}}/T_N,$$

где $T_N = T$ — общее время поступления N заявок.

12.

$$\epsilon = \sqrt{\frac{p(1-p)}{N}} \Phi^{-1}\left(\frac{\beta}{2}\right).$$

Используемые подпрограммы: 8.3 — экспонента, 8.7 — оценка вероятности.

алг канал

арг RO = $\Phi^{-1}(\beta/2)$, R1 = λ , R2 = μ , R3 = T_d , RD = r_0

рез RX = N , RA = $\bar{T}_{\text{ож}}$, RB = $K_{\text{п}}$, RC = p , RE = ϵ

нач R4 = N , R6 = T , R7 = T_k , R8 = $T_{\text{ож}}$, R9 = $T_{\text{п}}$, RC = λ , RC = μ ,

RA = $t_{\text{ож}}$

РАДИАНЫ

БВ RO = $\Phi^{-1}(\beta/2)$, R1 = λ , R2 = μ , R3 = T_d , RD = r_0

$N = 0$

1 R4 = 0

00. Сх (0)

01. хП4

$N_s = 0$

2 R5 = 0

$T = 0$
 $T_{\text{в}} = 0$
 $t_{\text{ож}} = 0$
 $T_{\Pi} = 0$
 ни
 экспонента (апр $RD = r$, $RC = \lambda$, рез $RX = t$)
 3 $R6 = 0$
 4 $R7 = 0$
 5 $R8 = 0$
 6 $R9 = 0$
 02. $x\Pi 5$
 03. $x\Pi 6$
 04. $x\Pi 7$
 05. $x\Pi 8$
 06. $x\Pi 9$
 7 $RC = R1$
 07. $\text{п}X1 (R1)$
 08. $x\Pi C (RC = R1)$
 8 GOSUB 35
 09. $\Pi\Pi$
 10. 86
 $T = T + t$
 9 $R6 = R6 + RX$
 11. $\text{п}X6 (R6, t)$
 12. $+ (t + R6)$
 13. $x\Pi 6 (R6 = T)$
 $N = N + 1$
 10 $R4 = R4 + 1$
 14. $K\Pi X4$
 экспонента (апр $RD = r$, $RC = \mu$, рез $RX = \tau$)
 11 $RC = R2$
 15. $\text{п}X2 (R2)$
 16. $x\Pi C (RC = \tau)$
 12 GOSUB 35
 17. $\Pi\Pi$
 18. 86
 $\tau = \text{знач}$
 13 $RC = RX$
 19. $x\Pi C (RC = \tau)$
 14 $RX = R6 - R7$
 20. $\text{п}X6 (R6)$
 21. $\text{п}X7 (R7, R6)$
 22. $- (R6 - R7)$
 15 IF $RX \geq 0$ ELSE 20
 23. $Fx \geq 0$
 24. 35
 то $T_{\Pi} = T_{\Pi} + t_{\Pi}$
 $T_{\text{в}} = T + \tau$
 16 $R9 = R9 + RX$
 25. $\text{п}X9 (R9, t_{\Pi})$
 26. $+ (t_{\Pi} + R9)$
 27. $x\Pi 9 (R9 = T_{\Pi})$
 17 $R7 = R6 + RC$
 28. $\text{п}X6 (R6)$
 29. $\text{п}X C (RC, R6)$
 30. $+ (R6 + RC)$
 31. $x\Pi 7 (R7 = T_{\text{в}})$

1	2	3		
		$N_s = N_s + 1$		
		18 $R5 = R5 + 1$		
		32. $K\Pi X5 (R5 = N_s + 1)$		
		19 GOTO 26		
		33. БП		
		34. 52		
		иначе $t_{\text{ож}} = T_{\text{в}} - T$		
		если $t_{\text{ож}} < T_{\Pi}$		
		21 $RX = R3$		
		39. $\text{п}X3 (R3, t_{\text{ож}})$		
		40. $- (t_{\text{ож}} - R3)$		
		22 IF $RX < 0$ ELSE 26		
		41. $Fx < 0$		
		42. 52		
		23 $R5 = R5 + 1$		
		43. $K\Pi X5 (R5 = N + 1)$		
		24 $R7 = R7 + RC$		
		44. $\text{п}X7 (R7)$		
		45. $\text{п}X C (RC, R7)$		
		46. $+ (R7 + RC)$		
		47. $x\Pi 7 (R7 = T_{\text{в}})$		
		$T_{\text{ож}} = T_{\text{ож}} + t_{\text{ож}}$		
		25 $R8 = R8 + RA$		
		48. $\text{п}X8 (R8)$		
		49. $\text{п}XA (RA, R8)$		
		50. $+ (R8 + RA)$		
		51. $x\Pi 8 (R8 + T_{\text{ож}})$		
		все		
		все		
кц	до ГРАДЫ			
	останов — ГРД (апр ГРАДЫ, рез останов)			
	26 SUB			
	52. 5	54. Ftg	56. $-$	58. 07
	53. 0	55. 1	57. $Fx = 0$	
	$\bar{T}_{\text{ож}} = T_{\text{ож}} / N$			
		27 $RA = R8/R5$		
		59. $\text{п}X8 (R8)$		
		60. $\text{п}X5 (R5, R8)$		
		61. $\div (R8/R5)$		
		62. $x\Pi A (RA = \bar{T}_{\text{ож}})$		
	$K_{\Pi} = T_{\Pi}/T$			
	$p = N_s/N$			

29 RC = R5/R4

67. пX5 (R5)
68. пX4 (R4, R5)
69. ÷ (R5/R4)
70. хПС (RC = p)

оценка вероятности (арг RO = $\Phi^{-1}(\beta/2)$, R4 = N, RC = p, рез RX = ε)
30 SUB

71. 1 73. — 75. X 77. + 79. пX0
72. пXC 74. пXC 76. пX4 78. FV 80. X

ε = знач

31 RE = RX

81. хПЕ (RE = ε)

ВЫ RX = N, RC = p, RE = ε, RA = $\bar{T}_{ож}$, RB = $K_{п}$
32 RX = R4

82. пX4 (R4)

33 STOP

83. С/П (N)

если продолжение вычислений;
то РАДИАНЫ; НЖ С/П;

34 GOTO 7

84. БП

85. 07

кон
все
35 SUB экспонента
86. пXD 89. X 92. K{х} 95. пXC 98. X
87. 1 90. Fπ 93. хПD 96. F1/x 99. В/O
88. 1 91. + 94. Fln 97. /—/

ИНСТРУКЦИЯ

BB RO = $\Phi^{-1}(\beta/2)$, R1 = λ, R2 = μ, R3 = T_d , RD = r_0
НЖ В/О С/П

выбор

останов вычислений: ГРАДЫ; ВЫ RX = N, RC = p, RE = ε, RA =
= $\bar{T}_{ож}$, RB = $K_{п}$
продолжение вычислений РАДИАНЫ; НЖ С/П

все

ТЕСТ

ГРАДЫ; BB RO = 1.96, R1 = λ = 100, R2 = μ = 1, R3 = T_d = 1, RD =
= r_0 = 0.5

НЖ В/О С/П; ВЫ RX = N = 1 (30 с), R5 = N_s = 1, RC = p = 1, RA =
 $\bar{T}_{ож}$ = 0, RB = $K_{п}$ = 1, RE = ε = 0
НЖ С/П; ВЫ RX = N = 2, R5 = N_s = 1, RC = p = 0.5, RA = $T_{ож}$ = 0,
RB = $K_{п}$ = 2.868821 — 01, RE = ε = 6.9296464 — 01

BB R3 = T_d = 100
НЖ С/П; ВЫ RX = N = 3, R5 = N_s = 2, RC = p = 6.6666666 — 01,
RE = ε = 5.3344439 — 1, RA = $T_{ож}$ = 8.0060995 — 01,
RB = $K_{п}$ = 2.5935094 — 01

8.6.3. ПРОВЕДЕНИЕ СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ НА МОДЕЛИ ОДНОКАНАЛЬНОГО НАПРАВЛЕНИЯ СВЯЗИ

Рассмотрим три случая:

интенсивность потока заявок вдвое меньше интенсивности потока обслуживания ($\lambda = 30$, $\mu = 60$ заяв./ч),

интенсивность потока заявок равна интенсивности потока обслуживания ($\lambda = 60$, $\mu = 60$ заяв./ч),

Результаты испытаний статистической модели одноканального направления связи

(начальное значение случайного числа ДСЧ $r_0 = 0.5$,
надежность оценки точности $\beta = 0.95$, $\Phi^{-1}(\beta/2) = 1.96$)

Исходные данные		Результаты					
заявок λ	обслуживания и	Интенсивность потока, заявок/ч	Число испытаний N	Вероятность обслуживания $p_{\text{т}} = p \pm \varepsilon$	Среднее время ожидания в очереди $T_{ож}$, ч	Коэффициент простой $K_{п}$	Время решения, ч
30	60	0	2134	0.694 ± 0.020	0	0.704	15
		0.05	2150	0.947 ± 0.009	0.016	0.490	15
		0.1	1184	0.949 ± 0.004	0.018	0.445	8
		0.5	1054	1.0 ± 0	0.018	0.445	8
60	60	0	1192	0.461 ± 0.028	0	0.474	8
		0.05	1828	0.778 ± 0.019	0.020	0.151	12
		0.1	1003	0.848 ± 0.022	0.044	0.085	7.5
		0.5	1051	0.932 ± 0.015	0.364	0.006	7.5
120	60	0	1060	0.312 ± 0.028	0	0.286	7.5
		0.05	1477	0.441 ± 0.021	0.374	0.003	10.5
		0.1	1788	0.444 ± 0.023	0.086	10	12.5
		0.5	1194	0.473 ± 0.028	0.455	10	8

интенсивность потока заявок вдвое больше интенсивности потока обслуживания ($\lambda = 120$, $\mu = 60$ заяв./ч). Для каждого из указанных случаев примем допустимое время ожидания в очереди на передачу сообщений $T_d = 0; 0.05; 0.1; 0.5$ ч. На разработанной статистической модели для каждой комбинации определим значения λ , μ , T_d , вероятность обслуживания p с оценкой ее точности ε при надежности оценки $\beta = 0.95$, среднее время ожидания заявки в очереди $T_{ож}$ и коэффициент простой $K_{п}$, показывающего, какую часть времени при заданных λ , μ , T_d канал простояивает. При этом для каждой комбинации исходных данных будем производить не менее 1000 статистических испытаний.

Результаты произведенных испытаний приведены в табл. 8.5. По указанным в п. 8.5.5 причинам полученные результаты приводятся в таблице с тремя знаками после десятичной точки.

Обратим внимание на случаи, когда допустимое время ожидания в очереди $T_d = 0$. Такие случаи имеют место, когда система массового обслуживания с очередью превращается в систему с отказами. Сравнивая результаты, полученные с помощью статистической модели (табл. 8.5) при $T_d = 0$ и аналитически (табл. 7.1 при $K_{п} = 0$), можно убедиться, что они не так сильно отличаются друг от друга (две — три единицы второго знака после десятичной точки). Различие полученных результатов объясняется тем, что аналитические формулы теории массового обслуживания

описывают установившийся стационарный режим работы системы и игнорируют наличие переходного процесса при начале ее функционирования. Действительно, в самом начале работы системы обслуживание заявок еще не начато и для первой поступившей заявки вероятность обслуживания всегда равна единице. Лишь после поступления серии заявок устанавливается режим с некоторой постоянной вероятностью обслуживания поступивших заявок. На время установления стационарного режима существенно влияет допустимое время ожидания заявки в очереди. В течение переходного процесса происходит формирование очереди сообщений, средняя длина которой в стационарном режиме остается постоянной. С увеличением допустимого времени ожидания, как это видно из табл. 8.5, вероятность обслуживания заявки повышается.

В методе статистических испытаний учитывается не только стационарный, но и переходный режимы систем массового обслуживания. Так как переходный процесс может длиться довольно долго, а иногда при реальной длительности работы системы стационарный режим может и не успеть установиться, результаты, полученные путем статистических испытаний на модели, могут оказаться более близкими к истине, чем аналитические расчеты по известным формулам. Кроме того, на статистических моделях можно получить и такие характеристики систем массового обслуживания, которые аналитически получить затруднительно. Например, немного изменив рассмотренную статистическую модель, можно вместо среднего времени ожидания получить среднюю длину очереди (среднее количество заявок, ожидающих обслуживания), учесть влияние надежности канала связи путем ввода в качестве исходного данного коэффициента готовности канала и др.

Возможности статистического моделирования на ПМК в действительности намного выше рассмотренных в данной главе. Можно построить еще более сложные модели, если третью их составную часть, предусматривающую заключительную обработку результатов серии произведенных испытаний, выполнять, используя ручной режим работы ПМК. Это вполне приемлемо, поскольку длительность автоматически выполняемых статистических испытаний составляет десятки часов, а обработка результатов серии испытаний на ПМК вручную — несколько минут. Но, что самое важное, такая обработка не искажает ни введенной программы, ни исходных данных, необходимых для продолжения статистических испытаний если число их окажется недостаточным.

В связи с массовой компьютеризацией, когда умение использовать ЭВМ в работе становится второй грамотностью, возникает вопрос о месте ПМК среди других средств вычислительной техники и возможности дальнейшего развития первых наряду со вторыми.

Программируемые микрокалькуляторы, как и персональные компьютеры (ПК), являются индивидуальными вычислительными средствами пользователя. Однако в отличие от ПК ПМК — автономные и переносные вычислительные средства, выполняемые конструктивно в виде моноблоков значительно меньших размеров.

Для того чтобы ПМК имели право на существование наряду с ПК, необходимо соблюдение ряда условий, состоящих в следующем:

1. Стоимость ПМК должна быть, по крайней мере, на порядок меньше стоимости простейшего персонального компьютера, рассчитанного на использование внешних устройств. Если в стоимость компьютера включать стоимость минимального комплекта внешних устройств, без которых он не может работать, то это условие легко выполняется уже в настоящее время, чему способствует отсутствие в ПМК:

устройства точной механики (это чисто электронное устройство);

внешних устройств, составляющих значительную часть стоимости ЭВМ.

2. Надежность ПМК (средняя наработка на отказ) должна быть на порядок выше надежности ПК. В настоящее время надежность ПМК значительно выше надежности выпускаемых ПК. В перспективе с повышением надежности используемых элементов (микросхем, индикаторов и др.) может быть существенно повышена и надежность ПМК в целом (в то время как надежность ПК ограничивается электромеханическими устройствами ввода-вывода).

3. Решение задач с использованием ПМК должно быть проще, удобнее, дешевле, быстрее, чем на ПК, существенной оговоркой: имеются в виду задачи того класса, для решения которых предназначены ПМК. Простота и удобство решения задачи обеспечивается тем, что ПМК может всегда находиться под рукой и не занимает много места. Низкая стоимость решения задачи достигается простотой схемы и низкой стоимостью самого ПМК. Быстрота решения обусловливается постоянной готовностью ПМК к работе и зависит от всех стадий разработки и решения задач, начиная от постановки, разработки алгоритма и программы и кончая отладкой, решением и сопровождением программы. Здесь быстродействие самого ПМК существенного значения не имеет. К сожалению, проблема обеспечения более высокой быстроты

(производительности) решения задач на ПМК еще ожидает своего решения. Однако если отказаться от стремления к бесконечному увеличению функций, выполняемых ПМК, то проблема повышения производительности решения задач с его помощью, по крайней мере до уровня ПК, в перспективе может быть решена. Первый шаг к решению этой проблемы — переход от бесскобочной записи выражений к алгебраической и применение в качестве входного языка программирования языка более высокого уровня, например БЕЙСИК-ПМК. Тогда вместо разработки алгоритма на трех уровнях его представления достаточно будет представить алгоритм только на двух уровнях: на лексиконе (исходный алгоритм) и на языке БЕЙСИК-ПМК, который будет одновременно служить и программой решения задачи, вводимой в память программируемого микрокалькулятора. Другим шагом в направлении ускорения решения задач на ПМК может служить облегчение и упрощение отладки программ путем отображения на индикаторе не кодов команд, как это делается сейчас, а текста самих команд. Это в перспективе возможно путем применения новых матричных алфавитно-цифровых жидкостно-криスタлических индикаторов и обеспечения во входном языке произвольной нумерации вводимых команд программы.

Выполнение перечисленных условий является необходимым, но не достаточным для существования ПМК как класса вычислительных средств (наряду с персональными компьютерами). С дальнейшим развитием микроминиатюризации электронных средств механические и электромеханические внешние устройства могут быть в перспективе заменены электронными, что даст возможность построить моноблоочную ЭВМ. В чем же тогда основное различие и основная причина параллельного развития персональных ЭВМ и ПМК как различных классов вычислительных средств?

Эффективность любого вычислительного средства определяется не столько его быстродействием, сколько временем ввода и подготовки исходных данных. Например, для того чтобы с помощью ЭВМ отыскать документ среди миллиона других, потребуется несколько минут, а для того чтобы ввести в память ЭВМ этот миллион документов, потребовалось бы около полумиллиона человеко-часов. Ускорение процесса ввода информации во всех типах ЭВМ, в том числе и персональных, достигается за счет использования для ввода информации пальцев обеих рук пользователя, что является одним из элементов компьютерной грамотности. При этом предполагается характер вводимой информации — текст, состоящий из ограниченного набора символов алфавита. Этим определяются минимальные размеры пульта управления ЭВМ.

Задача ускорения процесса ввода информации в ПМК решена по-иному. Здесь вместо ограниченного числа символов алфавита используются помимо цифр иероглифы, каждый из которых обозначает целый алгоритм (функцию, предписываемую к выполнению). Выполнение каждого иероглифа предписывается нажатием

одной клавиши. Количество иероглифов в принципе не ограничено, поэтому система команд практически любого ПМК содержит в своем составе большее их количество, чем система команд ЭВМ большой мощности. Пульт управления ПМК рассчитывается на использование одного пальца правой (левой) руки. Этим лимитируются минимальные габариты пульта управления ПМК, а значит, и всего ПМК. Достаточно выполнить эргономическое требование, чтобы палец руки не попадал на соседние клавиши. Способ ввода символов-иероглифов дает еще и то преимущество, что при соответствующей записи программ, как показывает опыт, практически исключаются ошибки ввода при переходе с регистра на регистр, которые составляют до 50 % при вводе информации с клавиатуры пульта ЭВМ.

Первый способ повышения скорости ввода информации наиболее эффективен для универсальных ЭВМ, второй — для специализированных вычислительных средств, поскольку количество клавиш, обозначающих иероглифы, всегда сильно ограничено. Отсюда следует, что основным направлением развития ПМК является их дальнейшая специализация. В настоящее время ПМК специализированы для обработки числовой информации вообще и адресованы инженерам и научным работникам всех специальностей. Более эффективными будут ПМК, ориентированные на автоматизацию вычислений, предусмотренных предметами, изучаемыми в средней школе, расчет электронных схем, а также электротехнических, геодезических и топографических, экономических и бухгалтерских расчетов. Возможно и целесообразно также создание ПМК, ориентированного на решение задач статистического моделирования систем массового обслуживания. Узкоспециализированные ПМК будут вне конкуренции при решении большинства часто встречающихся задач вычислительного характера (до 80 %).

В то же время решение более сложных задач вычислительного характера, построение различного рода математических моделей, создание обучающих систем, решение всех задач, связанных с обработкой текстов, могут быть выполнены только на ЭВМ более высокой производительности. Значит, в настоящее время и в перспективе необходимо и то и другое. Освоение ПМК дает возможность освоить основы компьютерной грамотности, так как подход к постановке задач, разработке алгоритмов, программированию и решению задач для ПМК и ЭВМ остается одним и тем же.

В настоящее время разработан и готовится к выпуску программируемый микрокалькулятор «Электроника МК-85», входным языком которого является язык БЕЙСИК, но более высокого уровня, чем описанный в данном пособии [4]. Служебные слова указанного языка вводятся в память ПМК * нажатием одной клавиши. Существенно упрощена и отладка программ. На ма-

* Аббревиатура ПМК в этом случае иногда расшифровывается как Переносной Микро Компьютер.

тричный жидкостно-кристаллический индикатор выдаются не только цифры, но и текст БЕЙСИК-команд программы. Хотя индикатор вмещает только одну строку с 12 символами, использование принципа бегущей строки дает возможность просмотреть строку, состоящую из 63 символов. Максимальная длина БЕЙСИК-программы — до 100 команд. Программная и регистрационная память объединены в общую оперативную память. В ней можно размещать как БЕЙСИК-команды, так и данные. Объем памяти составляет 2000 байт, из них 1000 байт доступны для пользователя. При выключении ПМК для хранения в памяти программ и данных требуется ток от автономных источников питания, соизмеримый с током их саморазряда. Пока батареи окончательно не разряжены, программа и данные сохраняются в памяти независимо от того, включен или выключен ПМК. При смене питания (четырех серебряноцинковых элементов СЦ-0.18, используемых в электронных часах) программа и данные в памяти ПМК сохраняются при отсутствии элементов до 15 мин. Длительность работы МК-85 от автономных источников питания — до 180 ч, что делает его практически полностью автономным.

Создание такого ПМК, как МК-85 показывает, что программируемые микрокалькуляторы наряду с персональными компьютерами представляют собой устойчивый класс вычислительных устройств, имеющих широкую перспективу своего дальнейшего развития.

ПРИЛОЖЕНИЕ

Обозначение клавиш различных ПМК типа «Электроника»

Выполняемые функции	Их обозначения на языке БЕЙСИК-ПМК	Обозначения клавиш		
		БЗ-34	МК-54, МК-56	МК-52, МК-61
Набор чисел:				
набор цифр числа	0—9	0—9	0—9	0—9
отделение целой части	.	,	.	.
числа от дробной				
брос неправильно	нет	Cx	Cx	Cx
введенного числа				
изменение знака числа	нет	/—/	/—/	/—/
на противоположный				
ввод порядка числа	E	BП	BП	BП
разделение одного	нет	↑	B↑(B!)	B↑(B!)
числа от другого при				
вводе серии чисел				
Выполнение одноместных операций				
$\sin x$	SIN (X)	Fsin	Fsin	Fsin
$\cos x$	COS (X)	Fcos	Fcos	Fcos
$\operatorname{tg} x$	TG (X)	Ftg	Ftg	Ftg
$\arcsin x$	ASIN (X)	Farcsin	Fsin ⁻¹	Fsin ⁻¹
$\arccos x$	ACOS (X)	Farcos	Fcos ⁻¹	Fcos ⁻¹
$\operatorname{arctg} x$	ATG (X)	Farctg	Ftg ⁻¹	Ftg ⁻¹
π	PI	Fπ	Fπ	Fπ
\sqrt{x}	SQR (X)	FV	FV	FV
x^2	$X * * 2$	Fx ²	Fx ²	Fx ²
$1/x$	$1/X$	F1/x	F1/x	F1/x
$\lg x$	LG (X)	Flg	Flg	Flg
10^x	$10 * * X$	F10 ^x	F10 ^x	F10 ^x
$\ln x$	LN (X)	Fln	Fln	Fln
e^x	EXP (X)	Fex	Fex	Fex
Выполнение двухместных операций				
сложение	+	+	+	+
вычитание	—	—	—	—
умножение	*	×	×	×
деление	/	÷	÷	÷
возведение в степень	* *	FX ^y	FX ^y	FX ^y
логическое сложение (или)	OR	нет	нет	K∨
логическое умножение (и)	AND	нет	нет	KΛ
исключающее или	XOR	нет	нет	K⊕
Обращение к ОЗУ данных:				
запись числа, индицируемого на экране, в регистр n :	Rn = RX	Pn	$x \rightarrow Pn$ (xPn)	$x \rightarrow Pn$ (xPn)

Выполняемые функции	Их обозначения на языке БЕЙСИК-ПМК	Обозначения клавиш		
		БЗ-34	МК-54, МК-56	МК-52, МК-61
считывание числа из регистра n и его индикация на экране	$RX = Rn$	ИПn	$\Pi \rightarrow xn$ (пXn)	$\Pi \rightarrow xn$ (пXn)
Управление вычислениями:				
обмен содержимым между регистрами X и Y;	$RY = RX$, $RX = RY$	\overleftrightarrow{XY} (XY)	\leftrightarrow	\leftrightarrow
вращение стека;	$RX = RY$, $RY = RZ$ нет	F	F (Fo)	F (F.)
восстановление результата предыдущего действия;		FBx	FBx	FBx
бросок нажатой префиксной клавиши;	нет	FCF	FCF	FCF
нет операции;	E (...)	КНОП нет	КНОП нет	К [x]
выделение целой части числа;	F (...)	нет	нет	K {x}
выделение дробной части числа;				
нахождение максимального числа из двух чисел, находящихся в регистрах X и Y;	MAX (...)	нет	нет	K_{\max}
определение абсолютного значения числа;	ABS (...)	нет	нет	K x
определение знака числа;	SIGN (...)	нет	нет	KZN
перевод градусов и минут угла в градусы и доли градуса;	MD (...)	нет	нет	K_o^{\rightarrow} (K,, $\rightarrow 0$)
перевод градусов и долей градуса угла в градусы и минуты;	DM (...)	нет	нет	K_o^{\leftarrow} (K,, $\leftarrow 0$)
перевод часов, минут и секунд в часы и доли часа;	MST (...)	нет	нет	K_o^{\leftrightarrow} (K,, $\leftrightarrow 0$)
перевод часов и долей часа в часы, минуты и секунды;	TMS (...)	нет	нет	K_o^{\leftrightarrow} (K,, $\leftrightarrow 0$)
генерация случайного числа;	RND	нет	нет	KСЧ
логическое отрицание	NOT	нет	нет	КИНВ

Выполняемые функции	Обозначение функций на языке БЕЙСИК-ПМК	Обозначения клавиш		
		БЗ-34	МК-54, МК-56	МК-52, МК-61
Управление работой программы:				
пуск или останов программы;	STOP	C/P	C/P	C/P
возврат счетчика адреса команд в начальное состояние или выход из подпрограммы для продолжения выполнения основной программы;	RETURN	B/O	B/O	B/O
обращение к подпрограмме;	GOSUB	ПП	ПП	ПП
безусловный переход к команде программы, адрес которой набирается вслед;	GOTO	БП	БП	БП
условный переход к команде программы при $X < 0$;	IF $RX < 0$ ELSE	$Fx < 0$	$Fx < 0$	$Fx < 0$
то же при $X = 0$;	IF $RX = 0$ ELSE	$Fx = 0$	$Fx = 0$	$Fx = 0$
то же при $X \geq 0$;	IF $RX \geq 0$ ELSE	$Fx \geq 0$	$Fx \geq 0$	$Fx \geq 0$
то же при $X \neq 0$;	IF $RX \neq 0$ ELSE	$Fx \neq 0$	$Fx \neq 0$	$Fx \neq 0$
организация счетчика в регистрах 0—3;	UNTIL $R0=1$ GOTO FL0	FL0	FL0	FL0
	UNTIL $R1=1$ GOTO FL1	FL1	FL1	FL1
	UNTIL $R2=1$ GOTO FL2	FL2	FL2	FL2
	UNTIL $R3=1$ GOTO FL3	FL3	FL3	FL3
просмотр кодов команд программы на экране индикатора.	нет	ШГ	ШГ	ШГ
сдвиг индицируемой команды вперед в сторону увеличения адресов команд;	нет	←	←	←
то же на шаг назад в сторону уменьшения адресов команд;	нет	ШГ	ШГ	ШГ
перевод ПМК в режим ввода программы;	FПРГ	FПРГ	FПРГ	FПРГ
перевод МПК в режим вычислений;	FABT	FABT	FABT	FABT
ввод адреса ППЗУ только для МК-52;	нет	нет	нет	A↑
запись, стирание, считывание программ и данных в ППЗУ (только для МК-52)	нет	нет	нет	↓↑

Примечание. В скобках даны упрощенные обозначения клавиш, используемые в литературе (в том числе и в данном пособии).

- 2.1.1. 1) 83 2) 0.01 3) -1 -02 4) -1 -015) 3.6280001 6) 12345678
 7) 0.1234567.8) -448.35 9) 0.125 -12 10) 421.65 -97 11) ЕГГОГ 12) 0.245 -99
 2.1.2. 1) 3.6280002 2) 1234567.9 -02. 3) 0.12345678 4) 9.1 -08 5) 1.2 -08
 6) 421.65 -97 7) 0. Примечание: в примере 6 необходимо масштабировать исходные данные, уменьшив их величину на порядок.
 2.2.1. 1) -5.3657277 -01 2) 7.0710681 -01 3) 5.7735027 -01 4) 1
 5) -5.1503808 -01 6) ЕГГОГ 7) 2.142422 рад; 122.75174° 8) 1.5703698 рад;
 89.97566° 9) 2.0523344 -12 10) 6.3254747 -01 11) 8.85481 -01 12) 6601.5625
 13) 3.1622775 14) 3.1622778 -0115) 25 16) ЕГГОГ 17) 4 18) 1 -09 19) 6.4461165
 20) 2.7995128 21) ЕГГОГ
 2.2.2. 1) 2.5833332 -01 = 0.25833332° 2) -30.209° 3) ЕГГОГ 4) 34.25°
 5) 22.175° 6) 1256.2782° 7) 63.0000166° 8) 1.0000016°
 2.2.3. 1) 0°21'.4" 2) -30°07'.5" 3) 16343°49.5' 4) 0°00.375'
 2.2.4. 1) 6.5383332 ч. 2) 3444.212 ч. 3) 0.0088888886 ч. 4) -124.01677 ч.
 2.2.5. 1) 11 ч. 43 мин. 30 с. 2) 0 ч. 00 мин. 22.5 с. 3) 156 ч. 00 мин. 00.1 с.
 4) -12 ч. 30 мин.
 2.3.1. 1) 529 2) 12267 3) 95431219 4) 1.274 -18 5) 59.01 6) 138.18 7) 3.1284567
 8) 2.788 -02
 2.3.2. 1) 69 2) -69 3) 12345597 4) 2.475 -18 5) -10.01 6) 51.404 7) 2.3384567
 8) -9.566 -04
 2.3.3. 1) 60592 2) 1.666355 -09 3) 4.5114074 -12 4) 9.84 -84 5) 706.75
 6) 9466.9127 7) 6.1530863 8) 1.468411 -23
 2.3.4. 1) 1.3984063 2) 5.3549514 -01 3) 85349.853 4) 2.6 -03 5) 4.005238
 6) 7.7303677 -01 7) 4.263393 8) 6.6288135 -02
 2.3.5. 1) 2.3474473 -83 2) 4095.9987 3) ЕГГОГ 4) 784685.14 5) 1.5009921
 6) 5.7744655 -21
 2.3.6. 1) 1.8387945 2) 9.79161 -03 3) 3.5000009 4) -2.00401 -01 рад
 или -11.482131° 5) 157.81999 рад или 246.20398° 6) 4.7415926 7) 3.7698986 -01
 2.4.1. 1) 111.9916 2) 153.4 3) 1026
 2.4.2. 1) 18.665266 2) 30.68 3) 146.57142
 2.4.3. 1) 120 2) 3628800
 2.4.4. 1) 4.0113153 2) 4.1864361 3) 8.6794871 -01; 4) 4.4241556 -01
 3.2.1. 1) $R_0 = 0.22 \cdot 5.65$; $R_0 = 8.58 \cdot 0.073 + R_0$; $R_1 = 24.3 \cdot 2.6$; $R_1 = 16.8 \cdot 3.3 + R_1$; $T = R_1/R_0 = 63.455551$; 2) $R_0 = 20^6$; $R_0 = \sqrt[3]{14964} + R_0$; $R_1 = 16^6$; $M = R_0 - R_1 = -13577188$; 3) $R_0 = 114 + 234 + 859 + 484$; $R_1 = 343 \cdot 458$; $R_1 = 507.9 \cdot 5 + R_1$; $R_1 = 387.174 + R_1$; $T = R_1/R_0 = 428.04671$
 3.2.2. 1) $R_0 = \varphi$; $R_1 = (1 - \tan^2 \varphi)/\tan^2 \varphi$; $R_1 = \sin^4 \varphi - R_1$; $P = R_1 + \cos^4 \varphi$
 ВВЕСТИ $R_0 = \varphi$
 00. 1 (1)
 01. $\pi X_0(\varphi, 1)$
 02. $F_{tg}(tg \varphi, 1)$
 03. $Fx^2(tg^2 \varphi, 1)$
 04. $- (1 - tg^2 \varphi)$
 05. $FBx(tg^2 \varphi, 1 - tg^2 \varphi)$
 06. $\div (1 - tg^2 \varphi)/tg^2 \varphi)$
 07. $x\Pi_1(R_1 = (1 - tg^2 \varphi)/tg^2 \varphi)$
 08. $\pi X_0(\varphi)$
 09. $F_{sin}(\sin \varphi)$
 10. $Fx^2(\sin^2 \varphi)$
 2) $P = 1.7216815 (\varphi = 72^\circ)$.
 3.2.3. $R_0 = 2.2^6$; $R_1 = \ln 2 \cdot \ln 3 + \ln 4 \cdot \ln 5$; $P = (\lg((3\sqrt{5} + 1/\sqrt{3}) - \arctg 1.2) \cdot 10^{-3}/R_1) - R_0)^2 + R_0 = 1465157.7$.
 3.2.4. $R_0 = K$; $R_1 = D$; $R_2 = \lambda$;
 1) ВВЕСТИ $R_0 = K$, $R_1 = D$, $R_2 = \lambda$;

00. $\pi X_1(D)$
 01. $\pi X_2(\lambda, D)$
 02. $\div (D/\lambda)$
 03. $F_{\Pi}(x, D/\lambda)$
 04. $\times (\pi D/\lambda)$
 05. $Fx^2((\pi D/\lambda)^2)$
 06. $\pi X_0(K, (\pi D/\lambda)^2)$
 07. $\times (K(\pi D/\lambda)^2)$
 08. $Flg(\lg K(\pi D/\lambda))$
 09. 1
 10. 0 (10, $\lg K(\pi D/\lambda)^2)$
 11. $\times (10 \lg K(\pi D/\lambda)^2)$
 ВЫДАТЬ G
- 2) $G = 36.243884 (K = 0.6, D = 0.8, \lambda = 0.03)$.
- 3.3.1. 1) 00. 2 (2) 02. $\times (2v)$ 04. $\cos(\cos \alpha, 2v)$
 01. $v(2, v)$ 03. $\alpha(\alpha, 2v)$ 05. $\times (2v \cos \alpha)$
- Бесскобочная запись: $2v \times \alpha \cos \times$.
- 2) 00. 2. 4 (2.4)
 01. $t(t, 2.4)$
 02. $\times (2.4t)$
 03. 5 (5, 2.4)
 04. $\pi(\pi, 5, 2.4)$
 05. $\times (5\pi, 2.4)$
 06. 18 (18, 5\pi, 2.4)
 07. : (5\pi/18, 2.4)
 08. $+(2.4t + 5\pi/18)$
 09. $\cos(\cos(2.4t + 5\pi/18))$
- Бесскобочная запись: $2.4t \times 5\pi \times 18 : + \cos 0.6 \times 1 + 4t \times \cos \times$.
- 3.4.1. Начальное состояние стека (1, 2, 3, 4):
00. F (2, 3, 4, 1)
 01. F (3, 4, 1, 2)
 02. F (4, 1, 2, 3)
 03. F (1, 2, 3, 4)*
 04. $\leftrightarrow (2, 1, 3, 4)$
 05. F (1, 3, 4, 2)
 06. F (3, 4, 2, 1)
 07. F (4, 2, 1, 3)
 08. F (2, 1, 3, 4)*
 09. F (1, 3, 4, 2)*
 10. $\leftrightarrow (3, 1, 4, 2)$
 11. F (1, 4, 2, 3)
 12. F (4, 2, 3, 1)
13. F (2, 3, 1, 4)
 14. F (3, 1, 4, 2)*
 15. F (1, 4, 2, 3)*
 16. F (4, 2, 3, 1)*
 17. $\leftrightarrow (2, 4, 3, 1)$
 18. F (4, 3, 1, 2)
 19. F (3, 1, 2, 4)
 20. F (1, 2, 4, 3)
 21. F (2, 4, 3, 1)*
 22. F (4, 3, 1, 2)*
 23. F (3, 1, 2, 4)*
 24. $\leftrightarrow (1, 3, 2, 4)$
 25. F (3, 2, 4, 1)
26. F (2, 4, 1, 3)
 27. F (4, 1, 3, 2)
 28. F (1, 3, 2, 4)*
 29. F (3, 2, 4, 1)*
 30. F (2, 4, 1, 3)*
 31. F (4, 1, 3, 2)*
 32. $\leftrightarrow (1, 4, 3, 2)$
 33. F (4, 3, 2, 1)
 34. F (3, 2, 1, 4)
 35. F (2, 1, 4, 3)
- Примечание. Всего $4! = 24$ различных комбинаций цифр 1, 2, 3, 4 в регистрах стека. Звездочкой помечены повторяющиеся комбинации.
- 3.5.1. 1) ВВЕСТИ 24.3!2.6 (2.6, 24.3)
00. $\times (63.18)$
 ВВЕСТИ 16.8 ! 3.3
 (3.3, 16.8, 63.18)
 01. $\times (55.44, 63.18)$
 02. $+(118.62)$
 ВВЕСТИ 0.22 ! 5.65
 (5.65, 0.22, 118.62)
 2) ВВЕСТИ 5 ! 20 (20, 5)
 00. FX^y (3199998, 2, 5)
 01. $\leftrightarrow (5, 3199998.2)$
 02. F (3199998.2)
 ВВЕСТИ 3 (3, 3199998.2)
 03. F1/x (3, 3333333 -01, 3199998.2)
 ВВЕСТИ 14964 (14964, 1/3, 3199998.2)
 04. FX^y (24.642373, 1/3, 3199998.2)
 05. $\leftrightarrow (1/3, 24, 642373, 3199998.2)$
 06. F (24.642373, 3199998.2)
 07. $+(3200022, 8)$

ВВЕСТИ 6 ! 16 (16, 6, 3200022. 8)
 08. FX^y (1677721, 1, 6, 3200022. 8)
 09. ↔ (6, 1677721. 1, 3200022. 8)
 10. F. (1677721. 1, 3200022. 8)
 11. — (-13577188)
 ВЫДАТЬ M = -13577188
 3) ВВЕСТИ 343 ! 458 (458, 343)
 00. × (157094)
 ВВЕСТИ 507 ! 985 (985, 507)
 01. × (499.395, 157094)
 ВВЕСТИ 387 ! 174 (174, 387, 499.395, 157094)
 02. × (67338, 499.395, 157094)
 03. + (566733, 157094)
 04. + (723827)
 ВВЕСТИ 114 ! 234 (114, 234, 723827)
 05. + (348, 723827)
 ВВЕСТИ 859 ! 484 (859, 348, 723827)
 06. + (1343, 348, 723827)
 07. + (1691, 723827)
 08. ÷ (428.04671)
 ВЫДАТЬ T = 428.04671
 3.5.2. 1) ГРАДУСЫ
 R0 = φ
 ВВЕСТИ R0 = φ
 00. πX0 (φ)
 01. Fsin (sin φ)
 02. Fx² (sin² φ)
 03. Fx². (sin⁴ φ)
 04. πX0 (φ, sin⁴ φ)
 05. Fcos (cos² φ, sin⁴ φ)
 06. Fx² (cos² φ, sin⁴ φ)
 07. Fx². (cos⁴ φ, sin⁴ φ)
 08. + (sin⁴ φ + cos⁴ φ)
 09. 1 (1, sin⁴ φ + cos⁴ φ)
 3.5.3. РАДИАНЫ:
 ВВЕСТИ 3 (3)
 00. F V (1.7320508)
 01. F1/x (5.7735027 -01)
 ВВЕСТИ 3 ! 5 (5, 5.7735027 -01)
 02. F V (2.2360679, 3, 5, 7735027 -01)
 03. × (6.7082037, 5.7735027 -01)
 04. + (7.285554)
 ВВЕСТИ 1.2 (1.2, 7.285554)
 05. Ftg⁻¹ (8.76058 -01, 7.285554)
 06. — (6.409496)
 ВВЕСТИ -3 (-3, 6.409496)
 07. F10^x (1 -03, 6.409496)
 08. × (6.409496 -03)
 ВВЕСТИ 2 (2, 6.409496)
 09. Fln (6.9314717 -01, 6.409496)
 ВВЕСТИ 3 (3, 6.9314717 -01, 6.409495)
 10. Fln (1.0986122, 6.9314717 -01, 6.49945)
 11. × (7.6149993 -01, 6.409495)
 ВВЕСТИ 4 (4, 7, 6.149993 -01, 6.409495)
 12. Fln (1.3862944, 7.6149993 -01, 6.409495)
 ВВЕСТИ 5 (5, 1.3862944, 7.6149993 -01, 6.409495)
 13. Fln (1.6094379, 1.3862944, 7.6149993 -01)
 14. × (2.2311547, 7.6149993 -01, 6.409495)
 15. + (2.9926546, 6.409495)

16. ÷ (2.1417426 -03)
 17. Flg (-2.6692327)
 ВВЕСТИ 9 ! 2.2 (2.2, 9, -2.6692327)
 18. FX^y (1207.269, 9, -2.6692327)
 19. ↔ (9, 1207.269, -2.6692327)
 20. F. (1207.269, -2.6692327)
 21. — (-1209.9382)
 22. FBx (1207.269, -1209, 9382)
 23. ↔ (-1209.9382, 1207.269)
 24. Fx² (1463950.4, 1207.269)
 25. + (1465157.7)
 ВЫДАТЬ P = 1465157.7
 4.2.1. 1) алг произведение p сомножителей * P = $\prod_{i=1}^n a [i] *$
 арг таб RX = a
 рез RX = P
 нач i, RY = P
 BB 1 (1) НЖ В!
 для i от 1 до длин (a)
 иц
 BB a [i] (a [i], P)
 P = P * a [i]
 ВЫ P
 кон
 2) алг площадь круга * S = πD²/4 *
 арг таб RX = D
 рез таб RX = S
 нач i
 для i от 1 до длин (a)
 иц
 BB D [i]
 S [i] = π * D [i] * * 2/4
 ВЫ S [i]
 кон
 3) алг факториал * n! (n ≠ 0) *
 арг RX = n
 рез RX = n!
 нач i, RY = N
 BB 1 (N = 1) НЖ В!
 для i от 1 до n
 иц
 BB i (i, N)
 N = N * i
 кон
 ВЫ RX = n! = N
 4.3.1. алг надежность * P = 1 - (1 - p)ⁿ *
 арг таб R0 = n, R1 = p
 рез таб RX = P [* , *]
 нач i, j
 для i от 1 до длин (P)
 иц
 BB R0 = n [i]

<p>1 2</p> <p>для j от 1 до шир (P) иц $BB R1 = p [j]$ $P [i, j] = 1 - (1 - p [j]) * * n [i]$</p> <p style="text-align: center;">НАЖАТЬ</p> <p>00. 1 (1) 01. пX1 (p, 1) 02. — ($1 - p$) 03. пX0 (n, $1 - p$) 04. ↔ ($1 - p$, n) 05. Fx^y ($(1 - p)^n$) 06. 1 ($1, (1 - p)^n$) 07. ↔ ($(1 - p)^n$, 1) 08. — ($1 - (1 - p)^n$)</p> <p>ВЫ P кц кон</p>	<p style="text-align: center;">Решение</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>x</th> <th>y</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1,7865624—01</td> </tr> <tr> <td>3</td> <td>2,6842853—01</td> </tr> <tr> <td>5</td> <td>3,039923 —01</td> </tr> </tbody> </table> <p>5.2.1. ПРОГРАММА расчета надежности системы из n элементов: алг надежность * $P = 1 - (1 - p) *$ арг таб $R0 = n$, $R1 = p$ рез таб $RX = P [* , *]$</p> <p>нач i, j для i от 1 до длин (P) иц $BB R0 = n [i]$ для j от 1 до шир (P) иц $BB R1 = p [j];$ НЖ В/О С/П $P = 1 - (1 - p [i]) * * n$ $1 RX = 1 - (1 - R1) * * R0$ 00. 1 (1) 01. пX1 ($R1$, 1) 02. — ($1 - R1$) 03. пX0 ($R0$, $1 - R1$) 04. ↔ ($1 - R1$, $R0$) 05. Fx^y ($(1 - R1) * * R0$) 06. 1 (1, $(1 - R1) * * R0$) 07. ↔ (($1 - R1$) * * $R0$, 1) 08. — ($1 - (1 - R1) * * R0$) 09. С/П</p> <p style="text-align: center;">2 STOP</p> <p>ВЫ P кц кон</p>	x	y	1	1,7865624—01	3	2,6842853—01	5	3,039923 —01
x	y								
1	1,7865624—01								
3	2,6842853—01								
5	3,039923 —01								
<p>ИНСТРУКЦИЯ</p> <p>для i от 1 до длин (P) иц $BB R0 = n [i]$ для j от 1 до шир (P) иц $BB R1 = p [j];$ НЖ В/О С/П ВЫ P кц</p>	<p>ТЕСТ $BB R0 = n = 2$, $R1 = p = 0,95$; НЖ В/О С/П; ВЫ $P = 9.975 —01$</p> <p>5.2.2. ПРОГРАММА расчета функции $Y = (A + Bx)/((A + x)(B + x));$ алг функция Y арг $R0 = x$, $R1 = A$, $R2 = B$ рез $RX = Y$</p>								

нач
 ВВ R1 = A, R2 = B; НЖК В/О С/П
 для x от 1 до 5 шаг 2
иц
 ВВ R0 = x

$$Y = (A + Bx)/((A + x)(B + x))$$

$$1 \text{ RX} = (R1 + R2 * R0)/((R1 + R0) * (R2 + R0))$$

$$\begin{aligned} &00. \text{nX1 (R1)} \\ &01. \text{nX2 (R2, R1)} \\ &02. \text{nX0 (R0, R2, R1)} \\ &03. \times (R2 * R0, R1) \\ &04. + (R1 + R2 * R0) \\ &05. \text{nX1 (R1, 04RX)} \\ &06. \text{nX0 (R0, R1, 04RX)} \\ &07. + (R1 + R0, 04RX) \\ &08. \text{nX2 (R2, 07RX, 04RX)} \\ &09. \text{nX0 (R0, R2, 07RX, 04RX)} \\ &10. + (R2 + R0, 07RX, 04RX) \\ &11. \times (07RX * 10RX, 04RX) \\ &12. \div (04RX/07RX) \end{aligned}$$

$$2 \text{ STOP}$$

$$13. \text{C/П (Y)}$$

$$\text{ВЫ Y}$$
кон
инструкция
 ВВ R1 = A, R2 = B
 для x от 1 до 5 шаг 2
иц
 ВВ R0 = x; НЖК В/О С/П

$$\text{ВЫ Y}$$
кон
тест ВВ R1 = A = 7.87, R2 = B = 10.75, R0 = x = 3; НЖК В/О С/П

$$\text{ВЫ Y} = 2.6842853 -01 (5 c)$$

5.2.3. 1) ПРОГРАММА $\sin x = (e^x - e^{-x})/2$:

алг sh x
 арг R0 = x
 рез RX = sh x

нач

ВВ R0 = x; НЖК В/О С/П

$$\text{sh } x = (e^x - e^{-x})/2$$

$$1 \text{ RX} = (\text{EXP (R0)} - \text{EXP (-R0)})/2$$

$$\begin{aligned} &00. \text{nX0 (R0)} \\ &01. \text{Fe}^x (\text{EXP (R0)}) \\ &02. \text{nX0 (R0, EXP (R0))} \\ &03. /-/ (-R0, EXP (R0)) \\ &04. \text{Fe}^x (\text{EXP (-R0)}, \text{EXP (R0)}) \\ &05. - (\text{EXP (R0)} - \text{EXP (-R0)}) \\ &06. 2 (2, 05RX) \\ &07. \div (05RX/2) \end{aligned}$$

$$2 \text{ STOP}$$

$$08. \text{C/П (sh } x)$$

$$\text{ВЫ sh } x$$
кон
инструкция ВВ R0 = x; НЖК В/О С/П; ВЫ sh x
тест ВВ R0 = x = 0.5; НЖК В/О С/П; ВЫ sh = 5.2109535 -01 (7 c)
2) ПРОГРАММА ch x = $(e^x + e^{-x})/2$:

алг ch x
 арг R0 = x
 рез RX = ch x
нач
 ВВ R0 = x; НЖК В/О С/П

$$\text{ch } x = (e^x - e^{-x})/2$$

$$1 \text{ RX} = (\text{EXP (R0)} - \text{EXP (-R0)})/2$$

$$\begin{aligned} &00. \text{nX0 (R0)} \\ &01. \text{Fe}^x (\text{EXP (R0)}) \\ &02. \text{nX0 (R0, EXP (R0))} \\ &03. /-/ (-R0, EXP (R0)) \\ &04. \text{Fe}^x (\text{EXP (-R0)}, \text{EXP (R0)}) \\ &05. + (\text{EXP (R0)} + \text{EXP (-R0)}) \\ &06. 2 (2, 05RX) \\ &07. \div (05RX/2) \end{aligned}$$

$$2 \text{ STOP}$$

$$08. \text{C/П (ch } x)$$
кон
инструкция ВВ R0 = x; НЖК В/О С/П; ВЫ ch x
тест ВВ R0 = x = 0.25; НЖК В/О С/П; ВЫ ch x = 1.0314131 (6 c)

3) ПРОГРАММА $\text{th } x = (e^x - e^{-x})/(e^x + e^{-x})$:

алг th x
 арг R0 = x
 рез RX = th x

нач

ВВ R0 = x; НЖК В/О С/П

$$\text{th } x = (e^x - e^{-x})/e^x + e^{-x}$$

$$1 \text{ RX} = (\text{EXP (R0)} - \text{EXP (-R0)})/(\text{EXP (R0)} - \text{EXP (-R0)})$$

$$\begin{aligned} &00. \text{nX0 (R0)} \\ &01. \text{Fe}^x (\text{EXP (R0)}) \\ &02. \text{nX0 (R0, EXP (R0))} \\ &03. /-/ (-R0, EXP (R0)) \\ &04. \text{Fe}^x (\text{EXP (-R0)}, \text{EXP (R0)}) \\ &05. - (\text{EXP (R0)} - \text{EXP (-R0)}) \\ &06. \text{nX0 (R0, 05RX)} \\ &07. \text{Fe}^x (\text{EXP (R0)}, 05RX) \\ &08. \text{nX0 (R0, EXP (R0), 05RX)} \\ &09. /-/ (-R0, EXP (R0), 05RX) \\ &10. \text{Fe}^x (\text{EXP (-R0)}, \text{EXP (R0)}, 05RX) \\ &11. + (\text{EXP (R0)} + \text{EXP (-R0)}, 05RX) \\ &12. \div (05RX/11RX) \end{aligned}$$

$$2 \text{ STOP}$$

$$13. \text{C/П (th } x)$$

$$\text{ВЫ th } x$$
кон
инструкция ВВ R0 = x; НЖК В/О С/П; ВЫ th x
тест ВВ R0 = x = 2; НЖК В/О С/П; ВЫ th x = 9.6402757 -01 (10 c)

5.5.1. ПРОГРАММА. Формула Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$,
 где $p = (a+b+c)/2$:

ВВ R0 = a	01. nX1	06. \div	11. \times	16. nX3	21. C/П
R1 = b	02. +	07. xP3	12. nX3	17. nX2	
R2 = c					
НЖК В/О С/П	03. nX2	08. nX3	13. nX1	18. —	
C/П	04. +	09. nX0	14. —	19. \times	
00. nX0	05. 2	10. —	15. \times	20. F ✓	

ТЕСТ ВВ R0 = $a = 0.25$; R1 = $b = 0.5$, R2 = $c = 0.6$
НЖ В/О С/П; ВЫ S = 6.1361505 -02 (10 c)

5.6.1. S = 2.4544602 -01, 7.6785389 -01, ЕГГОГ

6.1.1. ПРОГРАММА вычисления функции Y:

$$Y = \begin{cases} \sqrt{x} & \text{при } x \geq 0.5; \\ \cos x & \text{при } 0 < x < 0.5; \\ \sin x & \text{при } x \leq 0. \end{cases}$$

алг выбор
арг R0 = x
рез RX = Y

нач РАДИАНЫ
ВВ R0 = x; НЖ В/О С/П
выбор

при $x \geq 0.5$: $y = \sqrt{x}$
1 RX = R0 - 0.5

- 00. пX0 (R0)
- 01. 2 (2, R0)
- 02. F1/x (0.5, R0)
- 03. — (R0 - 0.5)

2 IF RX ≥ 0 ELSE 5
04. Fx ≥ 0
05. 10
3 RX = SQR (R0)
06. пX0 (R0)
07. F V (SQR (R0))

4 GOTO 11
08. БП
09. 21

при $x \geq 0$ и $x \neq 0$: $Y = \cos x$

5 RX = R0
10. пX0 (R0)
6 IF RX = 0 ELSE 10
11. Fx ≥ 0
12. 19
7 IF RX $\neq 0$ ELSE 10
13. Fx $\neq 0$
14. 19
8 RX = COS (R0)
15. пX0 (R0)
16. F cos (COS (R0))

9 GOTO 11
17. БП
18. 21

иначе $Y = \sin x$
10 RX = SIN (R0)
19. пX0 (R0)
20. F sin (SIN (R0))

11 STOP
21. С/П

все
Вы Y

кон
ИНСТРУКЦИЯ
РАДИАНЫ; ВВ R0 = x; НЖ В/О С/П; ВЫ Y
ТЕСТ

- 1) ВВ R0 = x = 0.8; НЖ В/О С/П; ВЫ Y = 8.9442719 -01 (3 c)
- 2) ВВ R0 = x = -0.8; НЖ В/О С/П; ВЫ Y = -7.173561 -01 (6 c)

3) ВВ R0 = x = 0; НЖ В/О С/П; ВЫ Y = 0 (5 c)

6.1.2. ПРОГРАММА вычисления $Y = \lg x$. При $x \leq 0$ положить $Y = -01$:

алг логарифм
арг RX = x
рез RX = Y

нач

ВВ x; НЖ В/О С/П
если $x \geq 0$ и $x \neq 0$

1 IF RX = 0 ELSE 5
00. Fx ≥ 0
01. 07
2 IF RX $\neq 0$ ELSE 5
02. Fx $\neq 0$
03. 07

то $Y = \lg x$
3 RX = LG (RX)
04. Flg (LG (RX))

4 GOTO 6
05. БП
06. 09

иначе $Y = 1.111111E-1$
5 RX = 1/9
07. 9 (9)
08. F1/x (1.111111 -01)

все
Вы Y

6 STOP
09. С/П (Y)

кон

ИНСТРУКЦИЯ ВВ x; НЖ В/О С/П; ВЫ Y
ТЕСТ

- 1) ВВ x = 2; НЖ В/О С/П; ВЫ Y = 3.0102999 -01 (4 c)
- 2) ВВ x = 0; НЖ В/О С/П; ВЫ Y = 1.1111111 -01 (2 c)
- 3) ВВ x = -2; НЖ В/О С/П; ВЫ Y = 1.1111111 -01 (2 c)

6.1.3. ПРОГРАММА вычисления $Y = \sqrt{-x}$. При $x > 0$ положить $Y = -01$:

алг корень
арг RX = x
рез RX = Y

нач

ВВ x; НЖ В/О С/П
если $x < 0$ или $x = 0$

1 IF RX < 0 ELSE 3
00. Fx < 0
01. 04

2 GOTO 4
02. БП
03. 06

3 IF RX = 0 ELSE 6
04. Fx = 0
05. 10

то $Y = \sqrt{-x}$

4 RX = SQR (-RX)
06. /-/

5 GOTO 7
07. F V

1

2

2

```

иначе  $Y = 1.1111111 - 01$ 
      6 RX = 1/9
      08. БП
      09. 12
      10. 9 (9)
      11. F1/x (1/9)
      12. С/П ( $\sqrt{x}$ )
      все ВЫ Y
      кон
      ИНСТРУКЦИЯ ВВ x; НЖ В/О С/П; ВЫ Y
      ТЕСТ
      1) ВВ x = 2; НЖ В/О С/П; ВЫ Y = 1.1111111 - 01 (2 с)
      2) ВВ x = -4; НЖ В/О С/П; ВЫ Y = 2 (3 с)
      3) ВВ x = 0; НЖ В/О С/П; ВЫ Y = 0 (2 с)

      6.2.1. ПРОГРАММА  $x = (1/i) \sum_{i=1}^i x [i]$ :
      алг среднее
          арг таб RX = x
          рез RX = x
          нач R0 = i, R1 = S
          i = 0
          1 R0 = 0
          00. 0 (0)
          01. xΠ0 (0)
          S = 0
          2 R1 = 0
          02. xΠ1 (0)
          для i от 1 до длин (x)
          иц
          3 STOP
          03. С/П (S [i - 1])
          ВЫ S [i - 1]
          BB x[i]; НЖ В/О С/П
          S = S + x [i]
          4 R1 = R1 + RX
          04. πX1 (R1, x [i])
          05. + (x [i] + R1)
          06. xΠ1 (R1 = S)
          i = i + 1
          5 R0 = R0 + 1
          07. 1 (1)
          08. πX0 (R0, 1)
          09. + (1 + R0)
          10. xΠ0 (R0 = i + 1)
          x = S/i
          6 RX = R1/R0
          11. πX1 (R1)
          12. πX0 (R0, R1)
          13. ÷ (R1/R0)
          7 GOTO 3
          14. БП
          15. 03
      кон
  
```

ИНСТРУКЦИЯ НЖ В/О С/П;
для i от 1 до длин (x)
иц
| ВВ RX = x [i]; НЖ С/П; ВЫ \bar{x}
кц
ТЕСТ НЖ В/О С/П (0);
ВВ x [1] = 1; НЖ С/П; ВЫ $\bar{x} = 1$ (3 с)
ВВ x [2] = 2; НЖ С/П; ВЫ $\bar{x} = 1.5$ (3 с)
ВВ x [3] = 3; НЖ С/П; ВЫ $\bar{x} = 2$ (3 с)

6.2.2. ПРОГРАММА $S = 1 + 1/x + 1/x^2 \dots$ при $x > 1$ с точностью 0.001:

```

алг сумма ряда
    арг R0 = x
    рез RX = S
    нач R1 = v, R2 = S
        BB R0 = x; НЖ В/О С/П
        v = 1
        1 R1 = 1
        00. КНОП
        01. 1 (1)
        02. xΠ1 (R1 = 1)
        S = 0
        2 R2 = 0
        03. 0 (0)
        04. xΠ2 (R2 = 0)
        иц
            S = S + v
            3 R2 = R2 + R1
            v = v/x
            4 R1 = R1/R0
            кц
                до v < 0.001
                5 RX = R1 - 0.001
                13. πX1 (R1)
                14. 0
                15. .
                16. 0
                17. 0
                18. 1 (0.001, R1)
                19. - (R1 - 0.001)
                6 IF RX < 0 ELSE 3
                20. Fx < 0
                21. 05
                ВЫ S
                7 RX = R2
                22. πX2 (S)
            кон
            ИНСТРУКЦИЯ
            BB R0 = x; НЖ В/О С/П; ВЫ S
            ТЕСТ
            BB R0 = x = 5; НЖ В/О С/П; ВЫ S = 1.2496 (25 с)
            6.2.3. ПРОГРАММА вычисления суммы натурального ряда чисел от 1 до n:
  
```

алг СМНР
 арг R1 = n
 рез RX = S
 нач R2 = i, R3 = S
 BB R1 = n; НЖ В/О С/П
 S = 0

1 R3 = 0
 для i от 1 до n
 2 R2 = 0
 3 R2 = R2 + 1
 4 RX = R1 - R2
 5 IF RX ≥ 0 ELSE 8
 10. Fx ≥ 0
 11. 18

НЦ S = S + i
 6 R3 = R3 + R2
 7 GOTO 3
 кон ИНСТРУКЦИЯ
 BB R1 = n; НЖ В/О С/П; ВЫ S

ТЕСТ
 BB R1 = n = 5; НЖ В/О С/П; ВЫ S = 15 (25 с)

6.3.1. Указанье. Диапазон изменения случайной величины в данном случае следует выбрать таким: M = 10 и m = 0. Тогда 1 попадает в первый интервал, 2 — во второй, ..., 6 — в шестой. Частичные интервалы 7, 8, 9, 10 не будут использованы для накопления частот попаданий в интервал.

7.2.1. ПОДПРОГРАММА NF1. N! с учетом 0! = 1:

алг NF1
 арг RX = N
 рез RX = N!
 нач R0 = N, RX = F
 если N $\neq 0$ то N = знач

1 IF RX $\neq 0$ ELSE 6
 00. Fx $\neq 0$
 01. 10

2 R0 = RX
 02. xΠ0 (R0)
 3 RX = 1
 03. 1 (F = 1)

НЦ F = F * N
 4 RX = RX * R0
 04. πX0 (R0, F)
 05. × (R0 * F)

КЦ до N = 1 от шаг -1
 5 UNTIL R0 = 1 GOTO 4
 06. FL0
 07. 04

6 GOTO 8
 08. БП
 09. 11

иначе F = 1
 7 RX = 1
 10. 1

все ВОЗВРАТ
 8 RETURN
 11. В/О

кон ТЕСТ
 1) BB RX = N = 4; НЖ В/О С/П; ВЫ N! = 24 (7 с)
 2) BB RX = N = 0. НЖ В/О С/П; ВЫ N! = 1 (2 с)

7.2.2. ПОДПРОГРАММА CNM1. $C_n^m = n!/(m!(n-m)!)$:

алг CNM1
 арг R1 = n, R2 = m
 рез RX = C_n^m
 нач R7 = F, pr R0

NF1 (арг RX = m, рез RX = m!, пр R0)
 1 RX = R2
 00. πX2 (R2)
 2 GOSUB 11
 01. ПП
 02. 19 (m!)

F = m!
 3 R7 = RX
 03. xΠ7 (R7 = m!)
 NF1 (арг RX = n, рез RX = n!, пр R0)
 4 RX = R1
 04. πX1 (R1)
 5 GOSUB 11
 05. ПП
 06. 19 (n!)

F = n!/F
 6 R7 = RX/R7
 07. πX7 (R7, n!)
 08. ÷ (n!/R7)
 09. xΠ7 (R7 = F)
 NF1 (арг RX = n - m, рез RX = (n - m)!, пр R0)
 7 RX = R1 - R2

1 2

8 GOSUB 11

$C_n^m = F/(n - m)!$

9 RX = R7/RX

ВОЗВРАТ

КОН

10 RETURN

11 SUB

12. — (R1 — R2)

13. ПП

14. 19 $((n - m)!)$

15. пX7 (R7, $(n - m)!$)

16. ↔ $((n - m)!, R7)$

17. $\div (R7/(n - m)!)$

18. B/O

19. Fx ≠ 0

20. 29

TECT

21. xP0

22. 1

23. пX0

24. ×

25. FL0

26. 23

27. БП

28. 30

29. 1

30. B/O

1 BB R1 = n = 6, R2 = m = 2; НЖК В/О С/П; ВЫ $C_6^2 = 15$ (25 с)

2 BB R1 = n = 6, R2 = m = 0; НЖК В/О С/П; ВЫ $C_6^0 = 1$ (25 с)

Преимущества: программа и при $m = 0$ будет давать правильный результат

$C_0^0 = 1$.

7.2.3. ПОДПРОГРАММА БРВ1:

$P_n^m = C_n^m p^m (1 - p)^{n-m}$ при $m \geq 0, n \geq 0, 0 \leq p \leq 1$

алг БРВ1

 апр R1 = n, R2 = m, R6 = p

 рез RX = P_n^m

нач pr R7

выбор

 при $p = 0: P_n^m = 0$

 1 RX = R6

 2 IF RX = 0 ELSE 4

 00. пX6 (R6)

 01. Fx = 0

 02. 05

 3 GOTO 18

 03. БП

 04. 56

 при $1 - p = 0$ и $n - m = 0: P = 0$

 4 RX = 1 — R6

 05. 1 (1)

 06. пX6 (R6, 1)

 07. — (1 — R6)

 5 RX = R1 — R2

 08. Fx = 0

 09. 18

 6 RX = R1 — R2

 10. пX1 (R1)

 11. пX2 (R2, R1)

 12. — (R1 — R2)

 7 IF RX = 0 ELSE 10

 13. Fx = 0

 14. 18

 8 RX = 1

 15. 1 (1)

9 GOTO 18

16. БП

17. 56

при $1 - p = 0: P_n^m = 0$

10 RX = 1 — R6

18. 1 (1)

19. пX6 (R6, 1)

20. — (1 — R6)

11 IF RX = 0 ELSE 13

21. Fx = 0

22. 25

12 GOTO 8

23. БП

24. 56

при $m = 0: P_n^m = (1 - p)^n$

13 RX = R2

25. пX2 (R2)

14 IF RX = 0 ELSE 17

26. Fx = 0

27. 35

15 RX = $(1 - R6) * * R1$

28. пX1 (R1)

29. 1 (1, R1)

30. пX6 (R6, 1, R1)

31. — (1 — R6)

32. Fx^y $((1 - R6) * * R1)$

16 GOTO 18

33. БП

34. 56

иначе БРВ — 7.3 (апр R1 = n, R2 = m, R1 = p,

 рез RX = P_n^m , пр R7)

17 SUB БРВ—7.3

35. пX1 40. — 45. пX2 50. FL1 55. ×

36. пX2 41. Fx^y 46. \div 51. 52

37. — 42. xP7 47. пX6 52. FL2

38. 1 43. 1 48. × 53. 44

39. пX6 44. пX1 49. × 54. пX7

все

ВОЗВРАТ

КОН

TECT

1) BB R1 = n = 4, R2 = m = 2, R6 = p = 0.25
НЖК В/О С/П; ВЫ $P = 2.1093749 -01$ (25 с)

2) BB R1 = n = 4, R2 = m = 4, R6 = 1
НЖК В/О С/П; ВЫ $P = 1$ (5 с)

3) BB R1 = n = 4, R2 = m = 2, R6 = 1
НЖК В/О С/П; ВЫ $P = 0$ (5 с)

4) BB R1 = n = 4, R2 = m = 2, R6 = p = 0
НЖК В/О С/П; ВЫ $P = 0$ (3 с)

5) BB R1 = n = 4, R2 = m = 0, R6 = p = 0.25
НЖК В/О С/П; ВЫ $P = 3.1640625 -01$ (10 с)

7.3.1. $n \ K_r \ \alpha \ K \ P_{обс}$

5	0.5	0.5	2	0.230769 -01
5	0.7	0.5	3	9.873418 -01

18 RETURN

56. B/O (P_n^m)

10 0.5 0.5 4 9.984202 —01
 10 0.6 1.0 6 9.99489 —01
 10 0.95 2.0 9 9.99809 —01

8.2.1. 1) ПОДПРОГРАММА ДСЧ1, $r_{i+1} = F(37r_i)$:

алг ДСЧ1
 арг RD = r_i
 рез RX = r_{i+1} , RD = r_{i+1}

нач

$r_{i+1} = F(37r_i)$

1 RD = F (37 * RD)
 00. пXD (RD)
 01. 3
 02. 7 (37, RD)
 03. × (37 * RD)
 04. K {x} (F (37 * RD))
 05. xΠD (RD = r_{i+1})

ВОЗВРАТ

2 RETURN
 06. B/O (r_{i+1})

кон

ТЕСТ

BB RD = $r_0 = 0.1234567$; НЖ В/О С/П; ВЫ 5.678979 —01 (2 с)
 НЖ В/О С/П; ВЫ 1.2222 —02

НЖ В/О С/П; ВЫ 4.52214 —01

2) ПОДПРОГРАММА ДСЧ2. $r_{i+1} = F(r_i/z_i + \pi)$, где $z_{i+1} = z_i + 10^{-8}$,
 $z_0 = 0.011$:
 алг ДСЧ
 арг RD = r_i , RE = z_0
 рез RX = r_{i+1} , RD = r_{i+1}

нач

$r_{i+1} = F(r_i/z_i + \pi)$

1 RD = E (RD/RE + PI)
 00. пXD (RD)
 01. пXE (RE, RD)
 02. ÷ (RD/RE)
 03. Fπ (RI, RD/RE)
 04. + (RD/RE+PI)
 05. K {x} (r_{i+1})
 06. xΠD (RD = r_{i+1})

$z_{i+1} = z_i + 10^{-8}$

2 RE = RE + 1E — 8
 07. пXE (RE)
 08. 8
 09. /—/ (-8, RE)
 10. F10^x (1E — 8, RE)
 11. + (RE + 1E — 8)
 12. xΠE (RE = z_{i+1})

знач = r_{i+1}

3 RX = RD
 13. пXD (r_{i+1})
 ВОЗВРАТ
 4 RETURN
 14. B/O (r_{i+1})

кон

ТЕСТ

BB RD = $r_0 = 0$, RE = $z_0 = 0.011$;
 НЖ В/О С/П; ВЫ 1.415926 (3.5 с)
 НЖ В/О С/П; ВЫ 1.3635 —02
 НЖ В/О С/П; ВЫ 3.811358 —01

8.3.1. ПРОГРАММА длина — ДСЧ. Определение количества неповторя-

ющихся случайных чисел, генерируемых ДСЧ:

алг длина — ДСЧ
 арг RD = r_i , RB = r_h * начальное число периодической части + 1 *
 рез RC = $N_{\text{пп}}$
 нач RC = i
 РАДИАНЫ; BB RD = r_0 , НЖ В/О С/П;
 $i = 0$
 1 RC = 0

00. Cx (0)
 01. xPC ($i = 0$)

ДСЧ1 (арг RD = r_0 , рез RD = r_i) *

* для примера взят ДСЧ1 с функцией преобразования $r_{i+1} = F(37r_i)$.
 В случае использования программы для проверки других ДСЧ соответственно
 должно быть заменено в скобках имя подпрограммы при обращении и ее текст
 (последовательность команд) *

2 GOSUB 11

02. ПП
 03. 31

$r_i = \text{знач}$

3 RB = RX

иц

4 RC = RC + 1

04. xΠB (RB = r_1)

ДСЧ1 (арг RD = r_i , рез RD = r_{i+1})
 5 GOSUB 11

09. ПП
 10. 31 (r_{i+1})

кц

до ГРАДЫ или $r_{i+1} = r_1$
 останов — ГРД (арг ГРАДЫ, рез останов)

6 SUB останов — ГРД
 11. КНОП 13. 0 15. 1 17. Fx = 0 19. пХС
 12. 5 14. Ftg 16. — 18. 21 20. С/П (i)
 7 RX = RB — RD

21. пXB (RB)
 22. пXD (RD, RB)
 23. — (RB — RD)

ВЫ $N_{\text{пп}}$

8 IF RX = RB — RD

24. Fx = 0
 25. 05

если продолжение
 то BB RB = r_h НЖ С/П

9 RX = RC

26. пХС (RC)

кон

10 STOP

27. С/П ($N_{\text{пп}}$)

11 SUB ДСЧ2

31. пXD 33. 7 35. K {x} 37. B/O
 32. 3 34. X 36. xΠD

28. БП

29. 05

ИНСТРУКЦИЯ
 НЖ БПЗ1 ГПРГ — ввести программу ДСЧ — ФАВТ

РАДИАНЫ; ВВ RD = r_0 ; НЖ В/О С/П
если ГРАДЫ
то останов; ВЫ RX = $i + 1$
все
ВЫ N_{ин} (>10 ч)

ТЕСТ

НЖ БП31 ФПРГ — ввести подпрограмму ДСЧ2 — FABT; ГРАДЫ
ВВ RD = $r_0 = 0.1234567$; НЖ В/О С/П; ВЫ $i = 1$ (8 с)

ВЫ RD = $r_2 = 1.2222 - 02$, RB = 5.678979 = r_1

РАДИАНЫ; ВВ RB = $r_3 = 7.31918 - 01$; НЖ С/П; ВЫ $i = 3$

ВЫ RD = $r_3 = 7.31918 - 01$

8.3.2. ПРОГРАММА — гистограмма — ДСЧ. Расчет гистограммы для ДСЧ.
Число интервалов гистограммы $n = 10$. Используются блоки программы 6.2 —
гистограмма. В качестве примера испытуемого ДСЧ взят простейший ДСЧ
с функцией преобразования $r_{i+1} = F(37r_i)$. В случае использования данной
программы для расчета гистограммы других ДСЧ соответственно должны быть
заменены имя подпрограммы ДСЧ при обращении к ней, а также ее текст (по-
следовательность команд).

Обозначения: i — текущее количество случайных чисел, выданных
ДСЧ и обработанных программой, j — номер интервала гистограммы, N —
заданное количество случайных чисел, подлежащих генерации ДСЧ, $N[j]$ —
количество случайных чисел, попадающих в j -й интервал гистограммы.

алг гистограмма — ДСЧ

арг RD = r_0 , R0 = N_3
результат RC = N_3 , R1 [10] = $N[10]$

нач RC = i , RB = j , пр RE

РАДИАНЫ

обнуление счетчиков (для j от 1 до 10 результат $N[j] = 0$)

1 SUB
00. Сх 03. 1 06. 0 09. 1 12. Fx = 0
01. хПВ 04. + 07. КхПВ 10. 0 13. 02
02. пХВ 05. хПВ 08. пХВ 11. —

$i = 0$

2 RC = 0
14. 0 (0)
15. хПС (RC = 0)

ВЫ π

3 RX = PI
16. F π (PI)
4 STOP
17. С/П (π)

ВВ RD = r_0 , R0 = N_3

НЖ С/П

иц ДСЧ1 (арг RD = r , результат RD = r_{i+1} пр RE)

5 GOSUB 15

18. ПП

19. 56

$i = i + 1$

6 RC = RC + 1
20. пХС (RC)
21. 1 (1, RC)
22. + (RC + 1)
23. хПС (RC = $i + 1$)

$j = E(r_i * 10 + 1)$

7 RB = E(10 * RD + 1)
24. пХД (RD)
25. 1 (1, RD)
26. 0 (10, RD)
27. X (10 * RD)

1

2

$N[j] = N[j] + 1$

8 R1 [RB] = R1 [RB] + 1

32. КпХВ (R1 [RB])

33. 1 (1, R1 [RB])

34. + (R1 [RB] + 1)

35. КхПВ [R1RB] = $N[j] + 1$

кц

до ГРАДЫ или $i = N_3$

останов — ГРД (арг ГРАДЫ, рез останов)

9 SUB

36. КНОП 38. 0 40. 1 42. Fx = 0 44. пХС

37. 5 39. Ftg 41. — 43. 46 45. С/П

10 RX = R0 — RC

46. пХ0 (R0)

47. пХС (RC, R0)

48. — (R0 — RC)

11 IF RX = 0 ELSE 2

49. Fx = 0

50. 18

ВЫ RX = $i = N_3$

12 RX = RC

51. пХС (RC)

13 STOP

52. С/П (N_3)

если продолжение
то НЖ С/П

14 GOTO 2

53. БП

54. 18

все

кон

15 SUB

56. пХД 58. 7 60. К {x} 62. В/О

57. 3 59. X 61. хПД

ИНСТРУКЦИЯ

РАДИАНЫ; НЖ БП 56 ФПРГ — набрать программу ДСЧ — FABT

ВВ RD = r_0 , R0 = N_3

НЖ В/О С/П; ВЫ π ; *π — признак обнуления интервальных счетчиков *

НЖ С/П

если останов

то ГРАДЫ; ВЫ i

если $i < N_3$

то РАДИАНЫ; НЖ С/П

все

ВЫ $i = N_3$, R1 = $N[1]$, R2 = $N[2]$, ..., Ri = $N[j]$, ..., RA = $n[10]$

Для продолжения вычислений ВВ R0 = $N_3 > N_3$; НЖ С/П

ТЕСТ

ГРАДЫ; НЖ БП 56 ФПРГ — набрать программу ДСЧ — FABT

ВВ RD = $r_0 = 0.1234567$, R0 = 5

НЖ В/О С/П; ВЫ π (35 с)

НЖ С/П; ВЫ RX = $i = 1$ (12 с), R6 = $N[6] = 1$

НЖ С/П; ВЫ RX = $i = 2$, R1 = $N[1] = 1$, R6 = $N[6] = 1$

НЖ С/П; ВЫ RX = $i = 3$, R1 = $N[1] = 1$, R5 = $N[5] = 1$, R6 = $N[6] = 1$

РАДИАНЫ; НЖ С/П; ВЫ RX = $N_3 = 5$

СПИСОК ЛИТЕРАТУРЫ

1. Алексеев В. Е., Баулин А. С., Петрова Г. Б. Вычислительная техника в инженерных и экономических расчетах. — М.: Высшая школа, 1984. — 136 с.
2. Астанин Л. Ю., Дорский Ю. Д., Костылев А. А. Применение программируемых микрокалькуляторов для инженерных и научных расчетов. — Л.: Энергоатомиздат, 1986. — 176 с.
3. Бойко А., Поташов А. Всего один диод//Наука и жизнь. — 1987. — № 4. — С. 123.
4. Бойко А., Чикоруди Р. Компьютер в кармане//Наука и жизнь. — 1987. — № 4. — С. 33—37.
5. Метод статистических испытаний (метод Монте—Карло)/Н. П. Бусленко, Д. И. Голенко, И. М. Соболь и др. — М.: ГИФМЛ, 1962. — 332 с.
6. Вентцель Е. С. Теория вероятностей. — М.: ГИФМЛ, 1962. — 564 с.
7. Гершензон Е. М., Полянина Г. Д., Соина Н. В. Радиотехника. — М.: Просвещение, 1986. — 320 с.
8. Гумрман В. Е. Теория вероятностей и математическая статистика. — М.: Высшая школа, 1977. — 500 с.
9. Голенко Д. И. Моделирование и статистический анализ псевдослучайных чисел на электронных вычислительных машинах. М.: Наука, 1965. — 228 с.
10. Горелик Г. С. Колебания и волны. — М.: ГИФМЛ, 1959. — 572 с.
11. Данилов И. Д. Секреты программируемого микрокалькулятора. — М.: Наука, 1986. — 160 с.
12. Дьяконов В. П. Расчет нелинейных и импульсных устройств на программируемых микрокалькуляторах. — М.: Наука, 1984. — 176 с.
13. Дьяконов В. П. Справочник по расчетам на микрокалькуляторах. — М.: Наука, 1985. — 224 с.
14. Ершов А. П., Монахов В. М. Основы информатики и вычислительной техники. Ч. 2. — М.: Просвещение, 1985. — 139 с.
15. Новиков О. А., Петухов С. И. Прикладные вопросы теории массового обслуживания. — М.: Сов. радио, 1969. — 400 с.
16. Кибернетика. Микрокалькуляторы в играх и задачах/Ю. В. Пухачев, В. А. Бардадым, Д. Д. Богданов и др. — М.: Наука, 1986. — 160 с.
17. Технико-экономические расчеты на программируемых микрокалькуляторах «Электроника»/под ред. Н. Н. Скворцова. — М.: Финансы и статистика, 1987. — 150 с.
18. Соболь Н. М. Метод Монте—Карло. — М.: Наука, 1985. — 78 с.
19. Трохименко Ф. К., Любич Ф. Д. Радиотехнические расчеты на микрокалькуляторах. — М.: Радио и связь, 1988. — 304 с.
20. Юропин Н. В. Геодезические вычисления на микрокалькуляторах и ЭКВМ. — М.: Недра, 1987. — 80 с.
21. Цветков А. Н., Епанечников В. А. Прикладные программы для микро-ЭВМ «Электроника Б3—34», «Электроника МК-54», «Электроника МК-56». — М.: Финансы и статистика, 1984. — 176 с.
22. Штильман З. М., Штильман Б. М. Метод программирования на БЕЙСИКе и ФОКАЛе на основе алгоритмического языка//Микропроцессорные средства и системы. — 1986. — № 3. — С. 34—38.
23. Штернберг Л. Ф. Курс программирования микрокалькуляторов Б3-34, МК-52, МК-54, МК-56, МК-61. — М.: Машиностроение, 1988. — 204 с.

ОГЛАВЛЕНИЕ

Предисловие	3
Раздел I. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРАХ И ВЫПОЛНЯЕМЫХ ИМИ ВЫЧИСЛИТЕЛЬНЫХ ОПЕРАЦИЯХ	
Глава 1. Принципы построения и функционирования ПМК	5
1.1. Технические требования и характеристики ПМК	6
1.2. Устройство ПМК	9
1.3. Функционирование ПМК при выполнении типовой команды	22
Глава 2. Основные вычислительные операции, выполняемые ПМК	25
2.1. Включение ПМК, ввод и индикация числа	29
2.2. Одноместные операции	36
2.3. Двухместные операции	41
Раздел II. ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ В РУЧНОМ РЕЖИМЕ	
Глава 3. Производство вычислений с использованием памяти ПМК	44
3.1. Команды обращения к регистрам адресуемой памяти	44
3.2. Использование адресуемой памяти для вычисления сложных выражений	47
3.3. Понятие об бескобоцкой записи выражений	51
3.4. Стековая память ПМК. Заполнение и выборка чисел из стека	55
3.5. Примеры вычислений с использованием стековой памяти	62
3.6. Рекомендации по использованию адресуемой и стековой памяти ПМК	64
Глава 4. Производство вычислений с многократно повторяющимися операциями	65
4.1. Алгоритм решения задачи. Язык описания алгоритмов	66
4.2. Примеры алгоритмов типовых вычислений	78
4.3. Пример решения задачи с представлением результатов в виде таблицы	81
Раздел III. ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ В ПРОГРАММИРУЕМОМ РЕЖИМЕ	
Глава 5. Составление линейных программ	88
5.1. Разработка алгоритма	90
5.2. Составление линейных программ	96
5.3. Ввод программы в память ПМК	101
5.4. Отладка программы	106
5.5. Оформление документации на программу	115
5.6. Решение задачи по документированной программе	119
Глава 6. Программирование ветвлений и повторений	121
6.1. Программирование ветвлений	133
6.2. Программирование повторений	146
6.3. Использование команд косвенной адресации	160
6.4. Особенности разработки и отладки сложных программ с ветвленими и повторениями	166
Раздел IV. РАЗРАБОТКА ПРОГРАММНЫХ КОМПЛЕКСОВ	
Глава 7. Общий порядок разработки программных комплексов	166
7.1. Вспомогательный алгоритм	166
7.2. Подпрограммы	168
7.3. Разработка программных комплексов из ранее составленных подпрограмм и программ	182

Г л а в а 8. Разработка программных комплексов для решения задач методом статистических испытаний	193
8.1. Понятие о методе статистических испытаний	196
8.2. Моделирование случайных величин	196
8.3. Обработка и выдача результатов статистических испытаний	211
8.4. Оценка точности результатов решения задачи по произведенному числу статистических испытаний	216
8.5. Пример построения на ПМК статистической модели усилительного каскада для оценки влияния разброса его элементов	220
8.6. Пример построения на ПМК статистической модели канала связи	232
Заключение (перспективы развития ПМК)	243
Приложение. Обозначение клавиш различных ПМК типа «Электроника»	247
Решения и ответы к упражнениям	250
Список литературы	270

ПРОИЗВОДСТВЕННОЕ ИЗДАНИЕ

Стрелянов Анатолий Иванович

ПРОИЗВОДСТВО ВЫЧИСЛЕНИЙ НА ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРАХ (МК-52, МК-54, МК-61)

Редактор Н. В. Сергеева

Переплет художника В. И. Коломейцева

Художественный редактор А. Н. Волкогонова

Технические редакторы Т. М. Жилич, А. И. Казаков

Корректоры И. Г. Иванова, А. И. Лавриненко

ИБ № 6744

Сдано в набор 29.08.89. Подписано в печать 22.03.90. М-31097. Формат 60×90^{1/16}. Бумага типографская № 2. Гарнитура литературная. Печать высокая. Усл. печ. л. 17,0. Усл. кр.-отт. 17,0. Уч.-изд. л. 20,52. Тираж 111 000 экз. Заказ 848. Цена 1 р. 80 к.

Ленинградское отделение ордена Трудового Красного Знамени издательства «Машиностроение». 191065, Ленинград, ул. Дзержинского, 10

Типография № 6 ордена Трудового Красного Знамени издательства «Машиностроение» при Государственном комитете СССР по печати. 193144, Ленинград, ул. Моисеенко, 10

