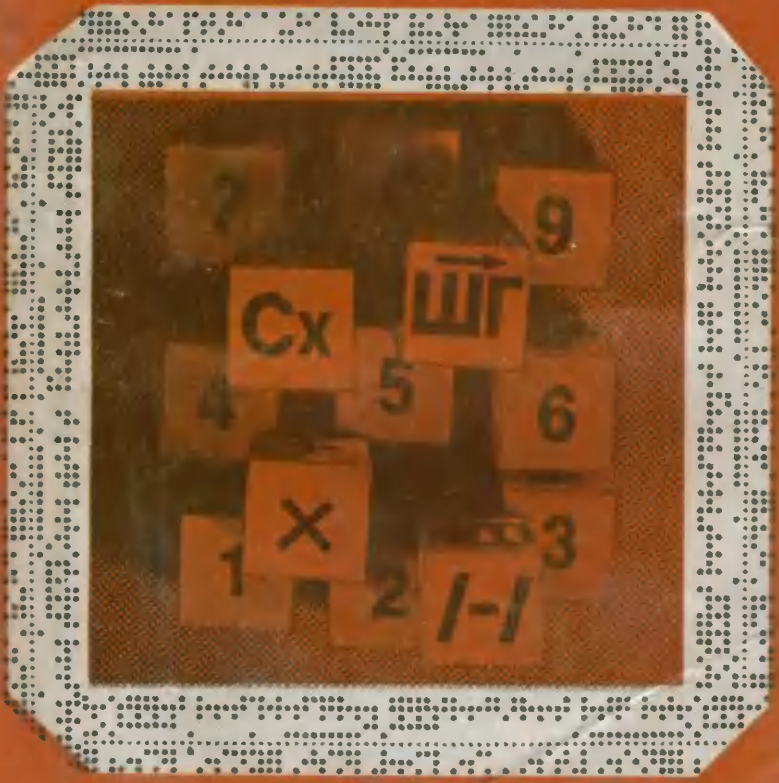
КИБЕРНЕТИКА

МИКРОКАЛЬКУЛЯТОРЫ В ИГРАХ И ЗАДАЧАХ



АКАДЕМИЯ НАУК СССР

СЕРИЯ «КИБЕРНЕТИКА —
НЕОГРАНИЧЕННЫЕ ВОЗМОЖНОСТИ
И ВОЗМОЖНЫЕ ОГРАНИЧЕНИЯ»

КИБЕРНЕТИКА

МИКРОКАЛЬКУЛЯТОРЫ В ИГРАХ И ЗАДАЧАХ

МОСКВА • «НАУКА»

1986

ББК 32.974
К 38
УДК 519.6

К 88 **Кибернетика. Микрокалькуляторы в играх и задачах.**— М.: Наука, 1986.— 160 с., ил.— (Серия «Кибернетика — неограниченные возможности и возможные ограничения»).

Сборник является своеобразным введением в информатику, показывающим возможности микрокалькуляторов. Занимательные игры позволят начинающему освоить приемы работы с микрокалькулятором, а разработка игровых программ познакомит с методикой программирования.

Для широкого круга читателей.

К $\frac{1502000000-402}{054(02)-86}$ 39—86 НП ББК 32.974

Серия «Кибернетика — неограниченные возможности и возможные ограничения» удостоена диплома I степени на Всесоюзном конкурсе общества «Знание» в 1985 г.

ПРЕДИСЛОВИЕ

На одном из заседаний по школьной информатике мне вручили папку с рукописью этой книги и попросили написать к ней предисловие. После заседания, раскрыв папку, я углубился в чтение.

Читал я книгу с нарастающим чувством сожаления. Сначала я пожалел, что у меня не оказалось под рукой микрокалькулятора, на котором я немедленно смог бы поиграть в предложенные игры.

Затем я не без зависти обнаружил, что сам не смог бы написать такую книгу, потому что ни опыт, ни знания сами по себе не могут заменить неисчерпаемого запаса выдумки и изобретательности, продемонстрированного авторами книги.

Потом я отметил, что неупорядоченный поток пусть мелких, но живых деталей и приемов, сопровождающий подробное описание процесса конструирования конкретной программы, иногда имеет определенное преимущество перед отпрепарированным дидактическими доктринами «анатомизированным» изложением разъятых правил систематического программирования.

После этого я очень пожалел, что игровой подход к обучению до сих пор остается сомнительной темой в глазах фундаментальной педагогики, опасливо дозирующей игру в пределах младших возрастных групп и «периферийных» предметов типа музыки и спорта.

Наконец я с грустью отметил, что сам уже никогда не угонюсь за ловкостью рук и быстротой мысли поколения молодых людей, которые становятся с компьютером «на ты» со школьных лет.

Если отмести все эти рефлексии, то можно сказать, что после знакомства с книгой остается ощущение интеллектуального напряжения и безграничного энтузиазма.

Возможно, микрокалькулятор не составил эпоху в историческом процессе компьютеризации общества — подобно тому, как планер не составил эпоху в практическом применении авиации.

Но точно так же, как полеты на планере стали неотъемлемой страницей биографии великих людей, составивших эпоху в развитии авиации, так и игра с микрокалькулятором послужит тем трамплином, который введет многих читателей этой книги в мир ЭВМ, в эпоху информатизации нашего общества.

Академик А. П. Ершов

ВВЕДЕНИЕ

Материал, содержащийся в книге, не требует от читателя особых познаний по вычислительной математике и навыков работы на ЭВМ. Он изложен простым и ясным языком и потому доступен самому широкому кругу читателей.

Степень сложности материала в книге нарастает от главы к главе. В первой собраны игры с микрокалькулятором, которые можно предложить тем, кто еще только начинает осваивать навыки счета. Но в таком случае работа на микрокалькуляторе должна проходить под руководством взрослых.

Игры, собранные во второй главе, уже не требуют помощи наставника.

Цель третьей главы — подготовить читателя к играм более увлекательным, в которых микрокалькулятор работает по определенной программе. Здесь объясняются приемы работы с программируемым микрокалькулятором «Электроника БЗ-34» и ему подобными.

В четвертой главе описаны две из наиболее популярных и в то же время простых игр с программируемым микрокалькулятором: «Посадка на Луну» и «Быки и коровы».

Программы более сложных игр приводятся в следующей, пятой главе. Они показываются в процессе их разработки.

В заключительной, шестой главе на примере игр с микрокалькулятором читатель знакомится с элементами теории игр.

От главы к главе меняется и форма изложения, чтобы наилучшим образом соответствовать их содержанию. Например, третья глава (она носит учебный характер) — это как бы ряд уроков по вычислениям на микрокалькуляторе; завершается она несложными задачами для самостоятельного решения. Пятая же глава написана в форме живого диалога, и такой выбор сделан не случайно: столь доходчивая форма позволяет лучше усвоить обилие собранного здесь материала.

Над книгой работали В. А. Бардадым, Д. Д. Богданов, А. Б. Бойко, И. Д. Данилов, С. И. Кадулин, Б. Н. Кульчицкий, Ю. В. Пухначев (руководитель авторского коллектива), Т. Б. Романовский, Г. В. Славин.

Глава 1

НАЧИНАЮЩИМ

Первая глава — первая ступенька к овладению микрокалькулятором. Она невысока — игры, собранные в этой главе, можно предложить даже малышам, которые только-только учатся считать.

Микрокалькулятор поможет им усвоить навыки счета, научиться цифровой записи чисел, натренировать память на числа, поупражняться в их сложении и умножении в уме.

Подготовить калькулятор к таким играм должны взрослые, неплохо владеющие программируемым микрокалькулятором «Электроника БЗ-34». Именно для него написаны приводимые ниже программы. С незначительными изменениями в обозначениях команд их можно переписать, чтобы играть по тем же правилам, для ряда других программируемых микрокалькуляторов (см. рис. 9).

ЦИФРЫ

Цифры, которые появляются на индикаторе микрокалькулятора, сильно отличаются по своему виду от тех, что мы пишем от руки, видим в книгах или на клавишах того же микрокалькулятора. Без умения читать эти стилизованные цифры невозможно освоить нашу карманную ЭВМ. Ребенок, который начинает знакомиться с нею, должен постичь соответствие между изображениями каждой цифры на индикаторе и на клавиатуре машины.

Помочь ему в этом призвана программа «Цифры». Задания «придумывает» сама машина. На индикаторе появляется какая-то случайная цифра, от 1 до 9. В ответ надо нажать на панели калькулятора соответствующую клавишу, на которой эта цифра изображена. Если задание выполнено неверно, то оно тотчас повторится вновь, и так будет до правильного выполнения. Если же нажата

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	Г√	2I	I4	ИПС	6C	28	↑	0E
0I	Г lg	I7	I5	−	II	29	↑	0E
02	ПД	4Г	I6	ПД	4Г	30	↑	0E
03	I	0I	I7	ИПС	6C	3I	↑	0E
04	0	00	I8	↑	0E	32	↑	0E
05	↑	0E	I9	С/П	50	33	↑	0E
06	9	09	20	−	II	34	↑	0E
07	:	I3	2I	Г x=0	5E	35	↑	0E
08	ПВ	4L	22	I7	I7	36	↑	0E
09	ИПД	6Г	23	⇄	I4	37	↑	0E
I0	Г $I0^x$	I5	24	ИПВ	6L	38	БП	5I
II	ПС	4C	25	×	I2	39	09	09
I2	КИПС	ГС	26	↑	0E			
I3	⇄	I4	27	↑	0E			

Рис. 1

правильная клавиша, то калькулятор отреагирует на это весьма красочным сообщением: цифра-задание будет восьмикратно повторена на индикаторе.

Например, в качестве очередного задания появилась цифра 4 и вслед за этим была нажата клавиша 4. В ответ на индикаторе загорится: 4,4444444. Это изображение будет мелькать несколько секунд, после чего появится новая цифра-задание.

Ребенку очень нравится такое мелькание. Но, к сожалению, эффектно оно выглядит лишь при вечернем освещении. В яркий день оно плохо заметно. Поэтому приводимый ниже текст программы снабжен примечанием о том, как изменить его, чтобы сообщение о правильном выполнении задания светилось долго, не мелькая.

Машина «придумывает» свои задания с помощью генератора случайных чисел (адреса 09—16), для запуска которого необходимо перед началом игры ввести в калькулятор какое-нибудь случайное число от 11 до 99. Это число должно быть нецелым, что-то вроде 28,3574 или

73,659 и т. п. Для получения такого числа вы можете взять цифры, которые выражают год, месяц и день вашего рождения, номер вашего дома и квартиры, номер своего телефона. Еще лучше придумывать случайное число так, чтобы оно всегда было новым. Удобно использовать для этого даты: например, день, в который вы играете в эту игру, часы и минуты, когда вы ее начинаете.

Инструкция.

1. Ввести программу (рис. 1).

2. Набрать случайное число. Нажать клавиши В/О С/П; на индикаторе — цифра. Нажать в ответ соответствующую цифровую клавишу, затем клавишу С/П. Прочитать на индикаторе повторение задания при неверном ответе или сообщение о правильном выполнении и затем новое задание.

Контрольный пример: 17,254 (случайное число) В/О С/П «4» 4С/П «4,4444444»...

Во всех инструкциях и описаниях контрольных примеров для программ, помещенных в этой главе, кавычки означают — читай на индикаторе, слова в скобках — пояснения.

Если ввести в микрокалькулятор приведенную программу, то сообщения о правильном выполнении заданий будут мелькать. Чтобы они светились ровно, окончание программы начиная с 26-го адреса должно быть иным: 26.С/П 27.БП 28.09. Прочитав сообщение о том, что задание выполнено верно, надо нажать клавишу С/П, и примерно через 5 с на экране появится новое задание.

Пояснения к тексту программы (по адресам):

00—02 — запоминание первого случайного числа в регистре Д;
03—08 — запоминание числа 1,1111111 в регистре В;
09—16 — генерирование случайной цифры от 1 до 9 в регистре С;
17—19 — индикация случайной цифры с засылкой ее в регистр Y;
20—22 — проверка правильности ответа, переход на повторный показ заданной цифры при неверном ответе;
23—25 — при верном ответе: пересылка цифры из регистра Y в регистр X, умножение ее на содержимое регистра В для формирования красочного сообщения о верном ответе;

26—39 — индикация сообщения о верном ответе на мерцающем индикаторе, после чего переход на генерирование новой случайной цифры.

ПОВТОРИ ЧИСЛО

С помощью этой программы ребенок научится вводить в машину и читать на индикаторе не только цифры, но и многоразрядные числа.

Машина «придумывает» задания при помощи генератора случайных чисел. В начале игры на индикаторе появляется какая-то цифра. Ее надо повторить нажатием на соответствующую цифровую клавишу, затем нажать клавишу С/П. По ходу игры задания усложняются, т. е. разрядность чисел растет.

Программа работает как своеобразный тренер — задания усложняются постепенно и только в случае успешного их выполнения. Если же ребенок начинает ошибаться, то разрядность чисел будет автоматически уменьшаться до такой степени сложности, покуда ошибки не исчезнут. Затем, после небольшой тренировки, разрядность опять начнет постепенно увеличиваться.

Программа принимает во внимание и случайные ошибки: если после долгого периода безошибочной работы вдруг появится ошибка, то она будет рассмотрена как случайная и число-задание просто повторится на индикаторе.

С помощью этой программы можно количественно оценивать привыкание ребенка к работе с многоразрядными числами, т. е. наблюдать, до скольких разрядов смог «доиграть» ребенок вчера, до скольких сегодня, и так в течение нескольких дней.

Так как программа регулирует степень сложности заданий автоматически, то ребенок, к своему удовольствию, может играть с микрокалькулятором и без участия взрослых.

Для запуска генератора случайных чисел перед началом каждой игры необходимо ввести какое-нибудь случайное нецелое число в интервале от 11 до 99, например, 73,4567.

Инструкция.

1. Ввести программу (рис. 2).

2. Случайное число, В/О, С/П «цифра или число»; повторить это набором с клавиатуры, С/П; . . .

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	FV	2I	I7	ИП5	65	34	БП	5I
0I	F1/x	I7	I8	FI0^{X}	I5	35	05	05
02	ПД	4Г	I9	×	I2	36	ИП4	64
03	Сх	0Г	20	ПС	4С	37	F x=0	5Е
04	П5	45	2I	КИПС	ГС	38	46	46
05	Сх	0Г	22	ИПС	6С	39	ИП5	65
06	П4	44	23	С/П	50	40	F x≠0	57
07	ИПД	6Г	24	-	II	4I	06	06
08	FI0^{X}	I5	25	F x=0	5Е	42	I	0I
09	ПД	4Г	26	36	36	43	-	II
I0	+	0Е	27	ИП4	64	44	БП	5I
II	КИПД	ГГ	28	F x=0	5Е	45	04	04
I2	FO	25	29	33	33	46	Сх	0Г
I3	ИПД	6Г	30	КИП4	Г4	47	П4	44
I4	-	II	3I	БП	5I	48	БП	5I
I5	ПД	4Г	32	07	07	49	22	22
I6	FC	25	33	КИП5	Г5			

Рис. 2

Время появления очередного задания на индикаторе — около 10 с.

Контрольный пример: набираем случайное число 73,4638 В/О, С/П «8»; 8 С/П «3»; 3 С/П «53»; . . .

Пояснения к программе (по адресам):

00—02 — подготовка и занесение в регистр Д первого случайного числа;

03—06 — очистка регистра 5 (регулятор разрядности, т. е. степени десяти) и регистра 4 (при правильном ответе туда заносится единица);

07—15 — генерация случайного числа;

16—23 — умножение случайного числа с целой частью на 10 в степени, равной содержимому регистра 5 (получение требуемой разрядности), вычисление целой части и ее индикация;

24—25 — проверка правильности ответа;
27—28 — проверка содержимого регистра 4 на равенство нулю;
30—32 — занесение единицы в регистр 4 (если там был нуль) и переход на генерацию нового задания без увеличения разрядности;
33—35 — увеличение разрядности в случае, если регистр 4 был не пуст (т. е. предыдущий ответ был правилен) и переход на засылку нуля в регистр 4 и на генерацию нового усложненного задания;
36—37 — проверка на равенство нулю содержимого регистра 4 при неправильном ответе;
39—45 — если при неправильном ответе содержимое регистра 4 равно нулю, то уменьшение разрядности на единицу с проверкой, чтобы не допустить отрицательного содержимого регистра 5 (39—40) и переход на новое, облегченное задание;
46—49 — если при неправильном ответе оказалось, что содержимое регистра 4 не равно нулю, то занесение нуля в регистр 4 и переход на повторный показ последнего задания.

МИНИ-СЧЕТ

Характер у этой программы — игровой, как и у предыдущих. И в то же время есть у нее учебный смысл: она обучает ребенка счету.

Задания, которые она дает малышу, двоякого рода. На индикаторе может появиться случайное количество «палочек» (т. е. единиц), которые надо сосчитать и нажать на соответствующую цифровую клавишу. В других случаях на индикаторе появляется случайная цифра — на сей раз в ответ следует ввести соответствующее ей число единиц. Интервал изменения цифр, как и числа «палочек»,— от 1 до 8.

Если задание выполнено неверно, то оно тотчас повторится на индикаторе. Если же оно выполнено правильно, об этом появится характерное сообщение. Например, если было задано ввести 6 палочек или подсчитать их количество в наборе 111111, то при верном ответе на индикаторе загорится: —666666. После появления этого сообщения нажатием клавиши С/П у калькулятора запрашивается новое задание.

Так как программа дает задания двух типов, то для перехода от одного к другому используется переклю-

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	Flg	I7	22	Cx	0Г	44	КИПС	ГС
0I	ПД	4Г	23	С/П	50	45	С/П	50
02	7	07	24	ИПД	6Г	46	ИПС	6С
03	П0	40	25	ИП2	62	47	-	II
04	I	0I	26	×	I2	48	Г x=0	5Е
05	ПI	4I	27	Гπ	20	49	44	44
06	ПС	4С	28	+	I0	50	БП	5I
07	$FI0^{x}$	I5	29	ПД	4Г	5I	59	59
08	ПА	4-	30	КИПД	ГГ	52	КИПС	ГС
09	ИПI	6I	3I	$\rightleftarrows$	I4	53	ИПС	6С
I0	ИПС	6С	32	ИПД	6Г	54	С/П	50
II	I	0I	33	-	II	55	КИПС	ГС
I2	+	I0	34	ПД	4Г	56	-	II
I3	ПС	4С	35	8	08	57	Г x=0	5Е
I4	$\rightleftarrows$	I4	36	×	I2	58	53	53
I5	ИПА	6-	37	I	0I	59	КИПС	ГС
I6	×	I2	38	+	I0	60	ИПС	6С
I7	I	0I	39	ПС	4С	6I	/-/	0L
I8	+	I0	40	Гπ	20	62	×	I2
I9	КПС	LC	4I	Гcos	IГ	63	БП	5I
20	FL0	5Г	42	Г x≥0	59	64	23	23
2I	I0	I0	43	52	52			

Рис. 3

чатель Р-Г. Если он находится в положении Р, то на индикаторе будут появляться цифры, и в ответ необходимо вводить требуемое число единиц. Если же переключатель находится в положении Г, то на индикаторе будут выстраиваться палочки, которые надо подсчитать и в ответ ввести соответствующую цифру.

Пользоваться переключателем можно в любой момент игры. Любое новое задание будет задано только после успешного выполнения предыдущего.

Свои задания машина подготавливает с помощью генератора случайных чисел. Для его запуска перед началом игры надо ввести какое-нибудь случайное число — любое нецелое число в интервале от 11 до 99, скажем 73,45234 и т. п.

Инструкция.

1. Ввести программу (рис. 3).

2. Набрать случайное число В/О С/П «О» (калькулятор готов к работе); С/П «задание»; дать ответ С/П «отметка о правильном выполнении или повторение в противном случае»; ... .

Время появления «О» — около 30 с, время появления очередного задания — около 9 с.

Контрольный пример: 54,321 В/О С/П «О»; Г С/П «11»; 2 С/П «—22»; С/П «111111» ...Р С/П «1»; 1 С/П «—1»; С/П «8»; 11111111 С/П...

Пояснения к программе (по адресам):

00—01 — подготовка и запись в регистр Д первого случайного числа;

02—23 — заготовка «палочек» в регистрах: 1 в регистре 1, 11 в регистре 2, ..., 11111111 в регистре 8; нуль на индикаторе;

24—39 — генерация случайного целого числа от 1 до 8 и его запись в регистр С;

40—43 — определение положения переключателя Р—Г, т. е. типа задания;

44—45 — индикация «палочек», соответствующих случайной цифре;

46—51 — проверка правильности подсчета палочек, переход на повтор задания при неправильном его выполнении (44) или на показ сообщения о правильности (59);

52—54 — индикация случайной цифры;

55—58 — проверка правильности ввода соответствующего количества палочек, переход на повтор задания при неправильном выполнении (53) или на показ сообщения о правильности (59);

59—64 — индикация сообщения о правильности выполнения задания и переход (23) на генерацию нового задания.

Эта программа предназначена для детей, уже вполне свободно владеющих вводом и чтением с индикатора многоразрядных чисел. Вообще говоря, программа может быть интересна детям любого школьного возраста и даже взрослым.

Заниматься этой игрой лучше всего вечером, так как числа, появляющиеся на индикаторе, горят недолго; их надо успеть прочесть и запомнить, читать же их легче при вечернем освещении. К тому же и сама игра требует спокойной, тихой обстановки, в которой можно лучше сосредоточиться.

Суть игры состоит в следующем. На индикаторе одно за другим появляются три случайных целых числа — обозначим их A, B и C. Каждое из них держится в течение примерно 1,5 с; для раздела между ними пару раз промелькнет шеренга из восьми единиц: 11111111. Затем эти числа требуется повторить набором с клавиатуры: A↑B↑C С/П. Если они были повторены неверно, то сообщение об этом появится в виде «—1,11111111», и задание повторится.

В начале игры числа будут однозначными — появятся три цифры. Затем задания станут усложняться: кромо того, что новые числа опять будут случайными, разрядность какого-то одного из них (также выбираемого случайно) увеличится на один разряд.

На своей начальной стадии игра доступна и посильна практически всем. Действительно, не так уж трудно запомнить и повторить три цифры. Но вот дойти до конца игры весьма непросто. Только люди с феноменальной памятью смогут повторить три промелькнувших 8-разрядных числа.

При этом можно количественно оценить свою память. Мерой тут может служить сумма разрядов, т. е. общее количество цифр в трех числах задания, которое явилось предельным для игрока. Эту оценку интересно сравнить с аналогичной оценкой памяти родственников, знакомых. Очень важно и то, что, часто играя в эту игру, можно заметить рост своей оценки. Неудивительно — память поддается тренировке.

В программе используется генератор случайных чисел, для запуска которого необходимо задать три случайных целых числа Для такой цели можно использовать дату, час и минуту текущего времени. Пусть, на-

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	+	I0	32	I	0I	64	-	II
0I	:	I3	33	+	I0	65	⇄	I4
02	ПД	4Г	34	КП↑	LE	66	ИП2	62
03	Сх	0Г	35	FLO	5Г	67	-	II
04	П5	45	36	I8	I8	68	+	I0
05	П6	46	37	КИПI	ГI	69	⇄	I4
06	I	0I	38	КИП2	Г2	70	ИПI	6I
07	0	00	39	КИП3	Г3	7I	-	II
08	П7	47	40	ИП5	65	72	+	I0
09	П8	48	4I	С/П	50	73	F x≠0	57
I0	П9	49	42	ИПI	6I	74	80	80
II	9	09	43	↑	0E	75	КИП6	Г6
I2	:	I3	44	↑	0E	76	ИПС	6С
I3	ПС	4С	45	↑	0E	77	/-/	0L
I4	3	03	46	↑	0E	78	БП	5I
I5	П0	40	47	ИПС	6С	79	4I	4I
I6	6	06	48	ИПС	6С	80	КИП5	Г5
I7	П4	44	49	ИП2	62	8I	ИПД	6Г
I8	ИПД	6Г	50	↑	0E	82	3	03
I9	I	0I	5I	↑	0E	83	×	I2
20	I·	0I	52	↑	0E	84	7	07
2I	×	I2	53	↑	0E	85	+	I0
22	Fπ	20	54	ИПС	6С	86	ПА	4-
23	+	I0	55	ИПС	6С	87	КИПА	Г-
24	ПД	4Г	56	ИП3	63	88	I	0I
25	КИПД	ГГ	57	↑	0E	89	0	00
26	⇄	I4	58	↑	0E	90	×	I2
27	ИПД	6Г	59	↑	0E	9I	КПА	L-
28	-	II	60	↑	0E	92	БП	5I
29	ПД	4Г	6I	ИПС	6С	93	I4	I4
30	КИП4	Г4	62	С/П	50			
3I	×	I2	63	ИП3	63			

Рис. 4

пример, сегодня, 13 мая 11 ч 35 мин. Набираем на клавиатуре 13↑11↑35.

После ввода этих трех чисел нажимаем В/О С/П и читаем на индикаторе «0». Это означает, что калькуля-

тор готов к игре и вместе с тем, что пока ни одно задание не выполнено: перед началом очередного задания будет показываться количество правильно выполненных. Внимательно смотрим на индикатор и нажимаем С/П. В течение 1,5 с промелькнет первое число, затем два раза мигнет 11111111, потом второе число, опять шеренга единиц и наконец третье число. После этого ярко высветится сигнал об окончании задания: 1,11111111. Теперь надо повторить три промелькнувших на индикаторе числа в той же последовательности, в какой они были показаны: А↑В↑С С/П.

В ходе игры в регистре 5 можно прочитать количество правильно выполненных заданий, в регистре 6 — количество выполненных неправильно.

Инструкция.

1. Ввести программу (рис. 4).
2. Дата↑час↑мин В/О С/П «0».
3. С/П «А...11111111...В..11111111...С..,
   ...1,1111111».
4. А↑В↑С С/П «количество выполненных заданий» (в случае правильного выполнения), после чего перейти к п. 3 и читать новое задание;
   «—1,1111111» (в случае неправильного выполнения), после чего — к п. 3 и читать повторение задания.
5. Для нового тура игры: БП 03 С/П «0» и перейти к п. 3.

Время появления «0» на индикаторе — около 25 с; время появления нового задания — около 30 с.

Контрольный пример:

13↑11↑35 В/О С/П «0»; С/П «9...11111111...8...
...11111111...2...1,1111111»; 9↑8↑2 С/П «1»; С/П...

Пояснения к программе (по адресам):

00—02 — подготовка и запись в регистр Д первого случайного числа;

03—13 — засылка нуля в регистр 5 (количество правильных выполнений заданий) и регистр 6 (количество неправильных), запись 10 в регистры 7, 8 и 9 (для увеличения разрядности каждого из трех задаваемых чисел), запись «1,1111111» в регистр С;

14—36 — цикл для генерации трех чисел, каждое из которых умножается соответственно на содержимое регистров 7, 8 и 9, после этого числа записываются в регистры 1, 2 и 3;

37—39 — выделение целой части трех чисел-заданий;
40—41 — индикация количества выполненных заданий;
42—62 — индикация подряд трех сгенерированных чисел в «мерцающем» режиме на индикаторе с «11111111» между ними (из регистра С);
63—74 — проверка на равенство трех показанных чисел и трех введенных (суммирование попарных разностей и проверка на равенство нулю этой суммы), переход на подготовку нового задания в случае равенства (80);
75—79 — в случае неправильного выполнения задания индикация сообщения «—1,1111111» и переход на повтор задания (41);
80—93 — подготовка к новому заданию с усложнением: преобразование случайного числа к интервалу от 7 до 9 (81—86) — так обеспечивается случайный вызов содержимого одного из регистров 7, 8 или 9 и увеличение его в 10 раз (88—91); затем переход на генерацию нового, усложненного задания (14).

УСТНЫЙ СЧЕТ

Когда ребенок начнет изучать арифметические операции, выучит таблицы сложения и умножения, когда ему захочется узнать, со сколь большими числами он способен совершать такие действия, ему можно предложить эту игру. Выигрыш в ней достигается искусством устного счета. Наверное, сыграть в нее будет интересно и взрослым.

Калькулятор предлагает играющему два числа. Их требуется сложить или перемножить. Выбор операции — дело вводящего программу. Если по адресу 27 записать команду +, то будут задаваться примеры на сложение, если команду × — то на умножение.

Переключателем Р-Г варьируется область предлагаемых чисел. Если поставить его в положение Р, то на индикаторе будут появляться как положительные, так и отрицательные числа, если в положение Г — то только положительные. Перевести переключатель из одного положения в другое можно в любой момент игры или перед пуском калькулятора.

Программа приспосабливается к уровню знаний играющего. Вначале на индикаторе появляются однознач-

ные числа. Потом их разрядность повышается, но если играющий начинает ошибаться, то снижается.

Рассмотрим конкретнее этот процесс приспособления. Предположим, что вы выбрали тот вариант игры, когда калькулятор задает примеры на сложение положительных чисел. Если до этого машинка была включена, выключите и включите ее снова, чтобы полностью очистить программную и числовую память. Введите программу, клавишей В/О установите счетчик адресов на нуль, переключатель Р-Г передвиньте в положение Г.

Нажмите клавишу С/П — на индикаторе через несколько секунд появится первое слагаемое, нажмите С/П — появится второе слагаемое. Подсчитайте сумму предложенных чисел, введите результат в регистр X, т. е. наберите подсчитанное вами число с помощью клавиатуры, нажмите С/П. После недолгой проверки калькулятор выставит оценку. Если расчет верен, то на индикаторе появится 5.5555555—ПП; цифры порядка ПП — это число верных ответов плюс единица. При новом нажатии клавиши С/П калькулятор предложит следующий пример. Если правильно решены пять примеров, калькулятор усложнит задание: начнет предлагать числа от 0 до 100. Так, постепенно усложняя задания, калькулятор доберется до примеров, справиться с которыми будет нелегко. Рано или поздно вы ошибетесь, и калькулятор отметит это сообщением 2.2222222—ДД; цифры порядка ДД — это число ошибочных ответов плюс два. Поскольку эта ошибка первая, калькулятор полагает ее случайной и после нажатия клавиши С/П предлагает тот же пример еще раз. Если ошибка действительно была случайной, калькулятор задаст следующий пример, а вот повторная ошибка послужит для него сигналом, что вы недостаточно готовы к решению задач такого уровня сложности, и он продиктует пять примеров с числами меньшей разрядности.

В программе используется датчик случайных чисел. Для его запуска требуется перед началом игры занести в регистр Д число в интервале от 0 до 1, например, 0, ЧЧММ, где ЧЧ — часы, ММ — минуты начала игры. Предположим, часы показывают 19 ч 15 мин. Вводим в регистр Д число 0.1915.

Инструкция.

1. Ввести программу, В/О (рис. 5).
2. Ввести О,ЧЧММ, ПД, С/П.
3. Читаем первое число, С/П.

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	В/О	52	33	П6	46	65	БП	5I
0I	4	04	34	I	0I	67	I2	I2
02	5	05	35	0	00	68	ИПД	6Г
03	F I/x	23	36	:	I3	69	5	05
04	П2	42	37	П5	45	70	×	I2
05	I	0I	38	С/П	50	7I	Fπ	20
06	,	0-	39	КИП4	Г4	72	+	I0
07	8	08	40	ИП4	64	73	↑	0E
08	F I/x	23	4I	5	05	74	ПД	4Г
09	П5	45	42	-	II	75	ВП	0[
I0	6	06	43	F x=0	5E	76	ПД	4Г
II	П0	40	44	I4	I4	77	Fπ	20
I2	Сх	0Г	45	FL0	5Г	78	F sin	IC
I3	П4	44	46	I3	I3	79	F x=0	5E
I4	ПП	53	47	В/О	52	80	87	87
I5	68	68	48	ИП6	66	8I	+	I0
I6	ПА	4-	49	F x≠0	57	82	2	02
I7	ПП	53	50	6I	6I	83	×	I2
I8	68	68	5I	Сх	0Г	84	I	0I
I9	ПВ	4L	52	П6	46	85	-	II
20	ИПВ	6L	53	ИП2	62	86	↑	0E
2I	С/П	50	54	I	0I	87	FO	25
22	ИПА	6-	55	0	00	88	ИП0	60
23	С/П	50	56	:	I3	89	F 10^x	I5
24	ПС	4C	57	П2	42	90	+	I0
25	ИПА	6-	58	С/П	50	9I	F Bx	0
26	ИПВ	6L	59	БП	5I	92	-	II
27	+	I0	60	20	20	93	7	07
28	ИПС	6C	6I	ИП0	60	94	ИП0	60
29	-	II	62	I	0I	95	-	II
30	F x=0	5E	63	+	I0	96	F 10^x	I5
3I	48	48	64	П0	40	97	×	I2
32	ИП5	65	65	П6	46			

Рис. 5

4. Читаем второе число, вводим результат операции С/П.

5. Читаем оценку, нажимаем С/П и переходим к п. 3.

Контрольный пример: вводим программу с коман-

дой + по адресу 27 (т. е. вариант с примерами на сложение), переводим переключатель Р-Г в положение Г (пример только с положительными числами) 0,1325 ПД В/О С/П «2» (1-е слагаемое) С/П «8» (2-е слагаемое) 10 (вводим подсчитанную в уме сумму) С/П «5.5555555—02» (расчет верен, 2—1=1, 1-й верный результат) С/П «9» (1-е слагаемое) С/П «10» (2-е слагаемое) 19 (вводим сумму) С/П «55555555—03» (расчет верен, 3—1=2; 2-й верный результат) ...С/П «24» (1-е слагаемое) С/П «82» (2-е слагаемое) 106 С/П «5.5555555—07» (расчет верен, 7—1=6; 6-й верный результат) С/П «90» (1-е слагаемое) С/П «35» (2-е слагаемое) 124 С/П «2.2222222—03» (расчет неверен, 3—2: 1-я ошибка) С/П «90» С/П «35» (повторная индикация слагаемых) 126 С/П «3» С/П «6» (повторная ошибка вызывает переход на группу примеров меньшей разрядности) и т. д.

Пояснения к программе (по адресам):

00—04 — формирование сообщения об ошибке;
05—09 — формирование сообщения о верном расчете 5.5555555—01;
10—11 — формирование признака работы с одноразрядными числами;
12—13 — очистка счетчика правильных ответов;
14—16 — генерация 2-го случайного числа;
17—19 — генерация 1-го случайного числа;
20—21 — индикация 2-го случайного числа;
22—23 — индикация 1-го случайного числа;
24 — запись предполагаемого результата;
25—27 — вычисление истинного результата;
28—31 — сравнение предполагаемого результата с истинным;
32, 34—37 — формирование сообщения о верном расчете и числе верных расчетов;
33 — запись признака отсутствия ошибки;
38 — индикация сообщения о верном расчете и числе верных расчетов;
39 — увеличение на единицу числа в счетчике верных ответов;
40—44 — проверка наличия пяти правильных ответов в вычислениях с числами текущей разрядности;
45—46 — увеличение разрядности;
47 — переход к началу программы после достижения максимальной разрядности, т. е. числа 10^7;
48—50 — контроль признака отсутствия ошибки;

51—52 — стирание признака отсутствия ошибки;
53—57 — формирование сообщения об ошибочном расчете и числе ошибочных расчетов;
58 — индикация сообщения о неверном расчете и числе неверных расчетов;
59—60 — переход на повтор примера, решенного с ошибкой;
61—64 — уменьшение разрядности;
65 — запись признака отсутствия ошибки;
66—67 — переход на серию примеров с числами меньшей разрядности;
68—76 — генерация псевдослучайного числа в диапазоне [0,1[по формуле $x_{i+1}=\{5x_i+\pi\}$;
77—80 — проверка режима, заданного с пульта (Р или Г);
81—86 — приведение псевдослучайного числа к диапазону [−1,1];
87—97, 00 — формирование целых положительных или отрицательных чисел заданной разрядности.

Глава 2

ИГРЫ БЕЗ ПРОГРАММ

Вторую ступеньку на пути к искусству вычислений с помощью микрокалькулятора читатель сможет осилить и без наставника. Правила игр, предлагаемых в этой главе, весьма просты и составлены так, что играть можно с непрограммируемым, т. е. простейшим или инженерным микрокалькулятором, скажем «БЗ-36» или «МК-51». Каждый ход в любой из этих игр заключается в том, чтобы набрать на цифровых клавишах какое-то число и нажать еще одну-две клавиши для выполнения тех или иных математических операций.

Во многих развлечениях с микрокалькулятором эта карманная ЭВМ подобна кубику Рубика или, скажем, коробке для игры в «пятнадцать».

Вторая из названных игр сегодня уже полузабыта. Напомним, как в нее играют. В плоской коробке располагаются 15 пронумерованных квадратных шашек, как показано на рис. 6. Задача состоит в том, чтобы посредством последовательных передвижений, допускаемых наличием свободного поля, перевести любое начальное расположение 15 шашек в нормальное, т. е. в такое, при

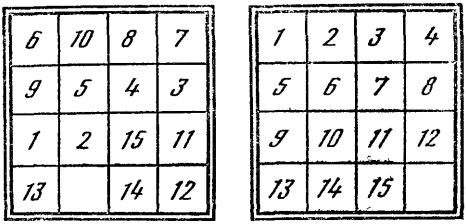


Рис. 6

котором шашки идут в порядке своих номеров: 1—4 в первом ряду, 5—8 во втором ряду и т. д.

Идея игры с кубиком Рубика по своей сути такова же. Из произвольной начальной мозаики посредством последовательных поворотов требуется получить кубик с одноцветными гранями.

По такому же принципу можно разработать целый ряд игр для микрокалькулятора. В каждой из них даются два числа — начальное и конечное. Путем последовательных математических действий, выполнимых на микрокалькуляторе, из начального числа требуется получить конечное. В зависимости от того, какие действия оговорены правилами игры (только сложение и вычитание, только извлечение квадратного корня и т. д.), получаем ту или иную игру.

КОНСТРУИРОВАНИЕ ЧИСЛА СЛОЖЕНИЕМ И ВЫЧИТАНИЕМ

Пусть начальное число равно, например, 17. Из него путем прибавления и вычитания двузначных чисел надо получить число 117.

Один из способов подобного «конструирования» числа 117 может быть таким. К исходному числу 17 прибавляем сначала 98. Для этого производим действия:

17+98=на индикаторе 115.

Число 115 меньше целевого числа 117. Прибавляем еще 14:

+14= на индикаторе 129.

Теперь в уме нетрудно оценить, что для получения требуемого числа надо вычесть из уже полученного 12:

−12=на индикаторе 117.

Число 117 сконструировано за три хода. А сможете ли вы сконструировать то же число за два хода?

У этой игры много возможных решений. Поэтому в ней могут участвовать много игроков. Попробуйте!

Как в любой игре, предполагается, что каждый из игроков честно соблюдает правила. Если забыть о них, то можно, конечно, взять произвольное число, например 78, затем с помощью калькулятора вычислить 17+78=95, а потом из 117 вычесть 95 и получить 22. Значит, число 117 можно конструировать в виде 17+78+22. Но это уже не игра. Если из коробки для игры в «пятнадцать» высыпать все шашки и потом их просто положить на свои места, то это тоже не игра. Другое дело, когда вы используете свои умственные способности. Их применение при конструировании чисел и делает игру интересной, а заодно развивает навыки вычислений в уме.

Теперь дадим задачи для упражнения.

Прибавлением и вычитанием двузначных чисел сконструировать:

из числа дней в неделе (7) число дней в году (365);

из года Великой Октябрьской социалистической революции (1917) год начала освоения космоса человеком (1961);

из среднего роста ребенка при рождении (50 см) свой теперешний рост.

Если в игре участвуют много игроков, то ход игры желательно протоколировать. Выигрывает тот, кто сконструировал число за наименьшее число ходов.

Конечно, в эту игру могут играть и дети младшего возраста. Ограничение относительно двузначных чисел в таком случае можно снять. Сперва детей надо ознакомить с игрой на таком, скажем, примере: из 4 сконструировать 13 прибавлением и вычитанием чисел, не больших 5:

$4 + 5 =$ на индикаторе 9
$+ 5 =$ на индикаторе 14
$- 1 =$ на индикаторе 13

Так пример за примером быстро вырабатываются навыки прикидочных вычислений в уме. Условия игры можно менять в соответствии со знаниями и умениями ребенка. Для детей можно предложить такие задачи:

из 3 сконструировать 10 прибавлением и вычитанием только чисел 2 и 3;

из 0 сконструировать 17 прибавлением и вычитанием только чисел 4 и 5 (или только 3 и 5);

из числа дней в неделе сконструировать число дней

в месяце (30, 28 или 31) прибавлением и вычитанием чисел 3 и 5;

из возраста одного из родителей сконструировать свой возраст в годах вычитанием и прибавлением 3 и 7.

КОНСТРУИРОВАНИЕ ЧИСЛА УМНОЖЕНИЕМ

Замена сложения и вычитания умножением усложняет конструирование числа. В этой игре вместо одного числа лучше указать интервал чисел. Например, пусть требуется из числа 12 сконструировать число от 137 до 139 включительно. Одно из возможных решений такое:

12×12 = на индикаторе 144
$\times 0{,}8$ = на индикаторе 115,2
$\times 1{,}1$ = на индикаторе 126,72

Дальше продолжайте сами, пока на индикаторе не появится число из интервала [137, 139].

Чем больше начальное число и чем уже целевой интервал, тем больше ходов требуется для конструирования числа. Попробуйте с помощью подобного конструирования, исходя из числа 19, попасть в интервал [94, 95], из 23 в [171, 173], из 937 в [402, 404], из 71 в [555, 556].

В эту игру можно играть и вдвоем, и в более широком кругу. Выигрывает тот, кто за меньшее количество ходов попал в заданный интервал.

Эта игра была придумана в США. Там она бытовала под названием «Испорченный микрокалькулятор» и формулировалась так: «У калькулятора не работает клавиша деления, а вам надо найти такой множитель X, который в произведении с заданным числом A давал бы другое заданное число B». Например, надо найти X такое, что $23 \cdot X = 851$.

Попытаемся решить задачу подбором. Пробуем:

23×40 = на индикаторе 920
23×35 = на индикаторе 805

Экспериментируя далее, можно вскоре определить, что X=37.

Тот вид, в котором эта игра была изложена здесь с самого начала, придал ей профессор Х. Майснер из ФРГ. Предложив свой вариант игры людям разных возрастов, он сделал немало любопытных наблюдений.

Так, например, 13-летнему школьнику, который не имел никаких навыков в подобных играх, профессор дал задачу: «Какое число надо умножить на 17, чтобы произведение находилось в промежутке от 560 до 585?». Решение ученик нашел только с 14-й попытки. Когда же он решил еще около 20 задач такого типа, Х. Майснер в качестве очередного задания поставил такое: «Найти число, которое в произведении с множителем 37 дает число от 960 до 965». С этой задачей ученик справился в три приема. Интересно, что никакие беседы о стратегии поиска решения с ним не проводились. Он сам разработал выигрышную стратегию и продемонстрировал при этом неплохое, как говорят педагоги, чувство числа.

Стоит заметить, что родители и учителя, протестующие против введения микрокалькуляторов в учебный процесс средней школы, нередко заявляют, будто школьники, увлекшись карманными ЭВМ, разучатся считать, утратят чувство числа. Приведенный пример опровергает это предвзятое мнение. Он показывает, что при умелом использовании микрокалькулятор, напротив, обостряет чувство числа — пусть не совсем в том виде, в котором оно понималось ранее.

Можно согласиться, что работа с микрокалькулятором несколько снижает навыки умножения многозначных чисел столбиком и их деления уголком. Но благодаря электронной вычислительной технике эти навыки сегодня обесцениваются точно так же, как механические часы заставили нас забыть искусство узнавать время по солнцу.

УГАДАЙ ЧИСЛО

Это популярная математическая игра, в которую без калькулятора играют так. Один игрок задумывает некоторое число, например 38, а другой должен его отгадать. Второй игрок называет некоторое число, например 20. В ответ он слышит: «Маловато». Допустим, второй игрок вслед за этим называет число 40. Ответ: «Многовато». И так до тех пор, пока число, названное вторым игроком, не совпадет с задуманным числом. Затем отгадывающий и загадывающий меняются ролями, и игра проводится вновь и вновь. Выигрывает тот, кто за меньшее число ходов отгадывает задуманное число.

Эту игру легко перенести на микрокалькулятор. Сперва рассмотрим такой ее вариант, при котором играюще-

му дается сильная подсказка. В этом случае задуманное число используется в качестве константы для автоматического вычитания. Примем, что первый игрок задумал число 387. Он набирает его на микрокалькуляторе, но так, чтобы второй игрок не видел, какое число набирается.

С 387 — 387 =на индикаторе 0 (для БЗ-36) или
С 387 — — =на индикаторе 0 (для МК-51)

Микрокалькулятор дается в руки второму игроку. Он пробует отгадать загаданное число и набирает, например,

500 = на индикаторе 113

Значит, 500 больше загаданного числа. Предположим, что далее второй игрок набирает

250 = на индикаторе −137

Это означает, что 250 меньше числа, задуманного первым игроком. Когда же второй игрок наберет

387 = на индикаторе 0

то появление нуля оповестит: число отгадано.

Эту игру также можно порекомендовать школьникам младших классов. Она развивает чувство числа, вырабатывает навыки сравнения чисел (больше, меньше, равно).

Для тех, кому легко даются вычисления в уме, отгадывание двух- или трехзначного числа не составит труда, поскольку его нетрудно найти по остатку вычитания. Например, когда на введенное число 250 калькулятор отвечает −137, то загаданное число быстро определяется суммированием в уме: 250+137=387. Для искусных вычислителей предлагаем вариант более сложной игры.

ЛАБИРИНТ

Рассмотрим цифровые клавиши, за исключением 0, как маршрутные точки для путешествия по клавишному полю (рис. 7).

Договоримся:

а) путешествие всегда начинается в точке 1 и кончается в 9;

б) в каждой точке маршрута можно побывать только один раз;

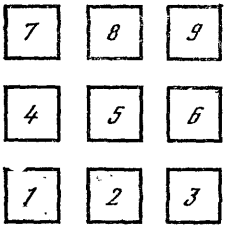


Рис. 7

в) в качестве следующей точки маршрута можно выбрать только близлежащую точку. Например, из «1» можно перейти на «4», «5» или «2», но нельзя переходить на «7» или «6».

Предположим, что вы как ведущий игры задаете маршрут 1—2—3—5—9. Подготовьте микрокалькулятор к игре:

С 12359 — 12359 = на индикаторе 0 (для БЗ-36) или
С 12359 — — = на индикаторе 0 (для МК-51)

Участнику игры надо отгадать ваш маршрут. Допустим, он набирает

14859 = на индикаторе 2500

Появление на индикаторе ненулевого числа означает, что маршрут не отгадан. Но по показанию индикатора видно, что предпоследняя точка маршрута «5» определена верно. Теперь можно пробовать другой маршрут с учетом новой информации:

12659 = на индикаторе 300

Результат всего лишь трехзначный. Стало быть, отгадана еще одна точка маршрута, клавиша «2»; теперь мыслимы только два возможных маршрута: 12459 или 12359. Набрано 12359, и на индикаторе появляется 0. Маршрут отгадан.

ИЗМЕРЕНИЕ ДЕЛЕНИЕМ

Игру «Угадай число» можно усложнить и другим способом — заменив сравнение чисел их делением. Предположим, что ведущий задумал число 38. Он готовит микрокалькулятор к автоматическому делению на 38:

С 38 : 38 = на индикаторе 1 (для БЗ-36) или
С 38 : : = на индикаторе 1 (для МК-51)

Микрокалькулятор передается второму игроку, который пробует

20 = на индикаторе 0,526.

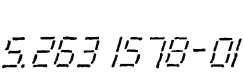
Так как на индикаторе число меньше единицы, то 20 — маловато. Игрок вводит большее число:

40 = на индикаторе 1,05...

Результат больше единицы, стало быть, 40 — многовато.

Когда игрок наберет 38 =, то на индикаторе появится 1. Число отгадано.

Попутно отметим одну деталь, которая может удивить начинающего. Если выполнить деление 20 на 38 с помощью разных микрокалькуляторов, то ответ 0,526... может появиться на индикаторе некоторых из них в не совсем привычной форме (выпишем полностью все знаки частного):



Расшифровывается эта форма следующим образом: выписывается число, знаки которого идут от левого края индикатора, и умножается на десятку в степени, равной числу, стоящему с правого края индикатора:

$5{,}2631578 \cdot 10^{-1} = 0{,}52631578.$

Это так называемая экспоненциальная форма представления чисел. Если калькулятор «владеет» ею, то так, в частности, в нем выводятся на индикатор числа, не превосходящие единицу по абсолютной величине.

Подробнее об экспоненциальной форме представления чисел будет рассказано далее, когда читатель станет знакомиться с программируемым микрокалькулятором «Электроника БЗ-34».

ОТЫЩИ КОРЕНЬ

До сих пор, играя с микрокалькулятором, мы использовали лишь арифметические действия: сложение и вычитание, умножение и деление. Между тем многие микрокалькуляторы, даже непрограммируемые, способны выполнять и более сложные операции: нахождение числа, обратного данному, вычисление синуса и косинуса от заданного числа, возведение числа в квадрат и т. д. Эти операции найдут свое применение в дальнейших играх, а пока в порядке разминки опробуем какую-нибудь из них.

Наберем на клавиатуре некоторое число, скажем 25, и нажатием клавиши x^2 возведем его в квадрат:

25 x^2 на индикаторе 625

На микрокалькуляторах, не имеющих клавиши x^2, возведение в квадрат можно проделать так:

$25\times$ = на индикаторе 625

Предположим, что у нас есть один из таких калькуляторов, не имеющий, впрочем, клавиши $\sqrt{}$. В случае такого неудобства корни можно отыскивать подбором, притом с любой точностью.

Требуется, например, отыскать $\sqrt{784}$. Иными словами, надо найти число, при умножении которого на себя получится 784.

Экспериментируем:

30 x^2 на индикаторе 900
27 x^2 на индикаторе 729
28 x^2 на индикаторе 784

т. е. искомый корень.

Интересно, за сколько ходов вы сможете найти квадратные корни из чисел 1369, 1936, 1024?

Справившись с этими задачами, попытайтесь найти приближенные значения квадратных корней из чисел 500, 700, 2000.

Эту игру можно предложить младшим школьникам, еще незнакомым с понятием квадратного корня. Игра поможет получить хорошее представление об этом понятии. Сначала же, разумеется, задачу надо ставить так: найти число, которое, будучи умножено само на себя, дает заданное число, например 169 или 361, 256 или 196.

ОТЫЩИ КОРЕНЬ ДРУГИМ СПОСОБОМ

Закройте одной рукой индикатор, а другой наберите

1369 $\sqrt{}$ — 1369 $\sqrt{}$ = (для БЗ-36) или
1369 $\sqrt{}$ — — = (для МК-51)

Снимите руку с индикатора. Если вы все сделали правильно, то на индикаторе должен быть 0. Теперь наберите пробное число, по вашему мнению, равное квадратному корню из набранного вначале числа, и нажмите клавишу =. Если на индикаторе положительный результат, то ваше число больше искомого корня, если отрицательный — то меньше. В случае успеха на индикаторе появится 0.

Подобным образом можно проверить навыки возведения числа в квадрат. Наберите, например,

23 x^2 — — = (для МК-51)

Оцените в уме квадрат числа 23, наберите его и нажмите клавишу =. Больше или меньше ваш ответ, нежели истинный?

Можно потренироваться в оценке обратной величины числа. Наберите

0,0016 $1/x$ — 0,0016 $1/x$ = (для БЗ-36) или
0,0016 $1/x$ — — = (для МК-51)

Нетрудно догадаться, что обратное значение числа 0,0016 меньше 1000. Наберите некоторое трехзначное число и нажмите клавишу =. Как близка ваша оценка к точному значению 1/0,0016?

Если вы желаете, чтобы калькулятор определял не абсолютную, а относительную погрешность ваших оценок, готовьте калькулятор к игре иначе. При отыскании корня:

С 1369$\sqrt{}$: 1369$\sqrt{}$ = (для БЗ-36) или
С 1369$\sqrt{}$: : = (для МК-51)

При возведении в квадрат:

С 23 x^2 : : = (для МК-51)

Далее игра идет так же, как в предыдущих вариантах. Если после вашего хода на индикаторе появится число, большее единицы,— ваша оценка завышена. Если меньшее — занижена.

СТРЕЛЬБА ПО ЦЕЛИ

Представьте себе, что у вас имеется пушка, стреляя из которой вам надо забросить трос помощи на маленький островок среди большого озера. Остров находится на расстоянии 0,650—0,665 км от пушки. Дальность полета определяется углом выстрела α и скоростью v, с которой из пушки вылетает метательный снаряд, увлекающий за собой трос.

Человек, знакомый с механикой, без труда определит, что дальность полета равна

$(v^2/g)\sin 2\alpha$.

Примем для простоты: скорость v, с которой вылетает снаряд из пушки, постоянна и притом такова, что дробь v^2/g равна единице. Тогда дальность стрельбы выражается величиной $\sin 2\alpha$ и отыскивается совсем просто: на-

бираем на клавиатуре удвоенное значение принятого угла стрельбы и нажимаем клавишу sin.

Установим для начала угол выстрела равным 10°:

20 sin на индикаторе 0,342

При таком значении угла трос упадет в озеро — недолет. Увеличим угол выстрела до 20°. Набираем

40 sin на индикаторе 0,642

Снова недолет — трос опять упадет в озеро. Дальше экспериментируйте сами. Если вам удалось найти угол выстрела, при котором трос упадет на остров, то попытайтесь найти еще одно значение угла, при котором трос также достигнет цели.

УДИВИТЕЛЬНЫЕ ЧИСЛА

Эту игру придумал Х. Гутцер из ГДР. На клавиатуре микрокалькулятора цифровые клавиши размещаются обычно так:

7 8 9
4 5 6
1 2 3

Составим четырехзначные числа, которые набираются по часовой или против часовой стрелки из цифр, расположенных в углах любого прямоугольника, образованного цифровыми клавишами. Например: 7931, 2365, 4631. Попробуем поделить эти числа на 11. Чтобы облегчить деление и каждый раз не набирать «делить на одиннадцать», подготовим микрокалькулятор к автоматическому делению:

С 11 : 11 = на индикаторе 1 (для БЗ-36) или
С 11 : : = на индикаторе 1 (для МК-51).

Теперь набираем указанные числа и производим деление на 11:

7931 = на индикаторе 721
2365 = на индикаторе 215
4631 = на индикаторе 421

Поразительно, но оказывается, что все эти числа делятся на 11 без остатка.

Продолжайте экспериментировать с другими числами: 1782, 3289 и т. д. Поищите числа, которые набираются указанным способом, но не делятся на 11.

В чем же секрет чисел, набиравшихся прежде и разделившихся на 11 без остатка? Калькулятор зовет пытливых к размышлению, к исследованию...

УПРЯМЫЙ КОСИНУС

Возьмите произвольное число, большее нуля и меньшее единицы. Представьте, что это радианная мера некоторого угла. Возьмите микрокалькулятор, переведите переключатель Р-Г в положение Р (радианы) и вычислите косинус взятого угла, нажав соответствующую клавишу. От полученного числа вновь возьмите косинус, с полученным на сей раз проделайте ту же операцию... и так далее.

Числа, последовательно загорающиеся на индикаторе, будут постепенно приближаться к некоторому числу.

Выберите другое число, большее нуля и меньшее единицы, и проделайте с ним ту же процедуру. Числа на индикаторе вновь будут приближаться к тому же пределу, что и прежде.

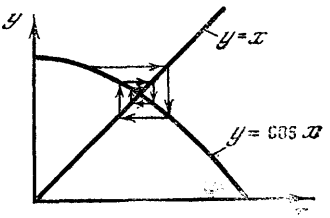


Рис. 8

Новые попытки убедят вас: каждый раз наблюдается стремление к пределу и этот предел не зависит от начального числа. Что же представляет собой этот предел? Какова его природа, его смысл?

Попытаемся разобраться в этих вопросах с помощью графика (рис. 8). На нем каждая вертикальная черточка, соответствующая n-му нажатию на клавишу cos, означает: вычисляется значение косинуса от аргумента x_{n-1}, горизонтальная: полученное число принимается за новое значение аргумента $x_n=\cos x_{n-1}$.

Стрелочки складываются в ломаную линию, спирально стягивающуюся к точке пересечения двух графиков: $y=x$ и $y=\cos x$. Это означает, что предел, к которому приближаются числа, появляющиеся на индикаторе, есть корень уравнения $x=\cos x$.

Вот как протекает этот сходящийся процесс, если за начальное приближение взять семь десятых:

$x_0 = 0{,}70000000$; $x_{11} = 0{,}73958447$;
$x_1 = 0{,}76484222$; $x_{12} = 0{,}73874869$.
$x_2 = 0{,}72149167$;

В пределе $x=0{,}739085$.

Метод, по которому в этих задачах отыскивается корень уравнения типа $x=f(x)$, называется методом итераций. Примеры его применения нетрудно умножить. Так с помощью вычислительных машин решаются не только отдельные уравнения, но и системы линейных и дифференциальных уравнений.

СИММЕТРИЧНЫЕ ЧИСЛА

Возьмите любое двухзначное число в интервале от 11 до 99. Прибавьте к нему зеркальное число, т. е. прочитанное справа налево исходное число. Например: 39+93=132.

К результату прибавьте его зеркальное число: 132+231=363. Вы получите симметричное число, т. е. такое, чье зеркальное число равно ему самому. В нашем примере для получения симметричного числа потребовалось два сложения.

Таким способом можно получить симметричное число из любого от 11 до 99. Для большинства чисел при этом потребуется от одного до четырех сложений. Лишь четыре числа в этом диапазоне несколько более упрямы — для получения из них симметричных чисел нужно шесть сложений. А два числа очень упорны — каждое из них превращается в симметричное лишь после 24 сложений! В результате получается симметричное число из 13 знаков.

Разрядная сетка нашего калькулятора вмещает всего 8 знаков, так что, если вы желаете получить такой результат, вам придется получать его вручную. Желаем успеха!

Для опытных любителей математических развлечений добавим, что эту игру можно вести и с трехзначными или даже с более длинными числами. Для подавляющего большинства чисел при этом удается достичь симметрии. Но есть несколько орешков, которые до сих пор не поддаются никаким усилиям, хотя над некоторыми уже произведено более тысячи сложений.

Математикам пока не удалось установить, для всех ли чисел можно получить симметричный итог путем конечного числа сложений, или существуют такие числа, которые для этой игры не годятся.

Для более увлекательных игр требуются калькуляторы классом повыше — те, что могут работать по вводимым в них программам.

Следующая глава будет посвящена знакомству с программируемым микрокалькулятором «Электроника Б3-34». Он пришел на смену первому отечественному программируемому микрокалькулятору «Электроника Б3-21» и за последние годы приобрел популярность наряду с построенными на основе тех же микросхем «Электроникой МК-56» и «Электроникой МК-54». Различаются эти три модели лишь некоторыми обозначениями на клавишах (см. рис. 9), поэтому программы, написанные для «Б3-34», годятся также для двух остальных моделей, и за некоторыми исключениями, о которых будет сказано в свое время,— для более мощных микрокалькуляторов «Электроника МК-52» и «Электроника МК-61».

Б3-34	МК-54 МК-56 МК-52 МК-61	В книге
ИП	$П \rightarrow x$	ИП
П	$x \rightarrow П$	П
$x \leftrightarrows y$	$\leftrightarrows$	$\rightleftarrows$
$\uparrow$	$\uparrow$	$\uparrow$
ШГ	ШГ	ШГ вправо
ШГ	ШГ	ШГ влево
÷	÷	:
$\circlearrowleft$	$\circlearrowleft$	F O
arcsin	$\sin^{-1}$	Farcsin
arccos	$\cos^{-1}$	Farccos
arctg	tg^{-1}	Farctg

Рис. 9

Глава 3

ШКОЛА ПРОГРАММИРОВАНИЯ

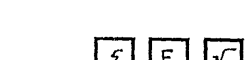
Эту главу можно было бы назвать «двухсполовинной» ступенькой к овладению карманными ЭВМ, потому что она носит учебный характер, помогает читателю освоить приемы работы с программируемым микрокалькулятором. Освоить их необходимо, чтобы приступить к более сложным и увлекательным играм, чем до сих пор (они описаны в следующих главах). Конечно, все эти приемы приведены в инструкции, прилагаемой к калькулятору. И все-таки для более прочного усвоения их нелишне повторить в популярной форме.

На панели калькулятора тридцать клавиш. На каждой проставлено свое обозначение. Кроме того, обозначения есть и над клавишами, а в нижнем ряду клавиатуры — и под ними. Такое обилие связано с тем, что наш калькулятор может выполнять довольно много разных операций (около двухсот). Поэтому каждая клавиша предназначена для выполнения двух, а то и трех действий.

Однако это не приводит к путанице. Опыт различения «что есть что» придет очень быстро.

Если нажать на клавишу, то будет выполняться действие, обозначенное на ней. Если же нажать сначала клавишу F и затем какую-то другую клавишу, то будет выполнена операция, обозначенная над клавишей.

Например, если надо вычислить $\sqrt{5}$, то нажимаем клавиши



На индикаторе загорается значение корня: 2,2360679. (Обратите внимание: числа, выводимые на индикатор вашего калькулятора, могут содержать до восьми разрядов.)

Отметим, что последняя из нажатых нами клавиш несет на себе символ «минус». Но вы, вероятно, не обратили на это внимание. Символ квадратного корня, написанный над нею, однозначно указал, что нажимать нужно было именно ее. Стало быть, вы уже приобретаете верную ориентацию в обозначениях на клавишах вашего калькулятора. В дальнейшем мы всегда будем называть требуемую по смыслу операцию независимо от того, обозначена она на самой клавише, сверху или снизу от нее.

Отметим и то, что для вычисления корня калькулятору сначала было сообщено подкоренное выражение (пять), а уже потом указано требуемое действие (хотя, казалось бы, традиционная запись корня $\sqrt{5}$ диктует иначе: сначала $\sqrt{}$, потом 5). Такой обратный порядок характерен для команд, отдаваемых нашему микрокалькулятору. К этому надо привыкнуть.

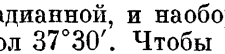
И еще примечание: действие клавиши F распространяется лишь на одну операцию, клавиша которой нажа-

та сразу вслед. Для повторного выполнения «надклавишной» операции (только что выполненной или любой другой) необходимо снова нажать клавишу F, а уже затем требуемую, скажем, lg или sin.

Бывает, что клавиша F нажата по ошибке. Ее действие отменяется клавишей CF. Нажмите ее, и если вслед за этим вы нажмете какую-то клавишу с обозначенной на ней операцией, то будет выполнена именно эта операция, а не та, что написана над клавишей.

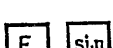
При вычислении тригонометрических функций надо отчетливо представлять себе, в каких единицах выражен угол — в радианах или в градусах. Соответственно надо устанавливать и переключатель Р—Г. При градусном измерении минуты и секунды передаются на индикаторе десятичной дробью. Например, 17°45′ будет выражено так: 17,75 (45′=0,75°).

С помощью переключателя Р—Г можно переходить от градусной меры к радианной, и наоборот. Например, переведем в радианы угол 37°30′. Чтобы ввести эту величину в калькулятор, превратим минуты в десятичную дробь: 37,5. Установим переключатель в положение Г, вводим это число последовательным нажатием клавиш



Заметьте: если число содержит не более восьми знаков, то ввести его проще всего именно так, как мы только что поступили — набирая на клавиатуре его последовательные цифры и не забывая своевременно нажать на клавишу «запятая», если это нужно.

Возьмем от введенного числа функцию «синус», нажав клавиши



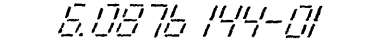
На индикаторе читаем значение синуса: 0,60876144.

Переставим переключатель в положение Р и от результата предыдущей выкладки возьмем арксинус:

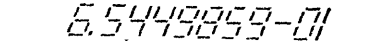
На индикаторе читаем радианную меру нашего угла: 0,65449859.

Впрочем, так прочтет показания индикатора лишь тот, кто обладает известной сноровкой в обращении с

выводимыми на него числами. На самом деле индикатор показывал сначала



а потом



Числа, по абсолютной величине меньшие 1 и большие 99999999, в микрокалькуляторе представляются в так называемой экспоненциальной форме (как еще говорят, с плавающей запятой). Такой вид числам придается с помощью степеней десятки:

$10^1=10$, $10^3=1000$, $10^7=10000000$, $10^{-1}=0{,}1$, $10^{-7}=0{,}0000001$ и т. д.

Любое число, как бы мало или велико оно ни было, можно представить в виде произведения двух сомножителей. В одном из них — те же цифры, что и в исходном числе, но запятая стоит сразу после первой ненулевой (значащей) его цифры. Этот сомножитель называется мантиссой. Другой сомножитель — это 10 в некоторой степени (она называется порядком). Например:

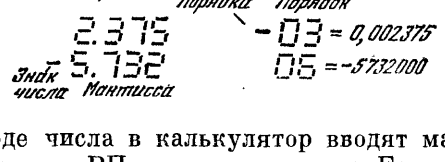
$23790000=2{,}379\cdot10^7$, $0{,}00002379=2{,}379\cdot10^{-5}$.

Использование такого представления, кроме чисто технических соображений, удобно во многих отношениях: благодаря этому калькулятор может работать воистину с гигантским диапазоном чисел — от 10^{-99} до почти 10^{100}. (Чтобы оценить по достоинству широту этого диапазона, примите во внимание такие цифры: масса протона — около 10^{-24} г, а масса Вселенной оценивается в 10^{55} г.)

На индикаторе порядок числа представлен двумя крайними правыми цифрами со знаком перед ними (+ не показывается). Все, что левее,— мантисса. Полученное в предыдущем примере число радиан перепишем с индикатора так:

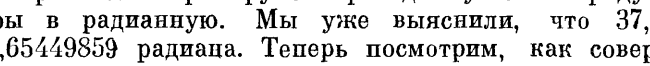
К такому представлению чисел привыкнуть совсем нетрудно. Для начала можно пользоваться таким сове-

том: если знак порядка отрицательный, то перед первой цифрой мантиссы напишите столько нулей, сколько указано в порядке, и поставьте запятую после первого нуля; если же порядок задан положительным числом, то на такое число цифр переносите запятую вправо — по исчерпании цифр мантиссы добавляйте нули:



При вводе числа в калькулятор вводят мантиссу, нажимают клавишу ВП и вводят порядок. Если число отрицательное, то после ввода мантиссы нажимают клавишу /—/. Если порядок отрицательный, то /—/ нажимают после его ввода.

Потренируйтесь сами в вводе чисел в экспоненциальной форме и умении их легко читать. Для этого используйте клавишу ↑; она автоматически нормализует набранное на клавиатуре число, меньшее единицы:



Вернемся к примеру с переводом угла из градусной меры в радианную. Мы уже выяснили, что 37,5° = =0,65449859 радиана. Теперь посмотрим, как совершается перевод из радианной меры в градусную.

Установив переключатель в положение Р, наберем 0,65449859 и нажмем клавиши F sin. На индикаторе читаем 0,60876151. Установим затем переключатель в положение Г и нажмем клавиши F arcsin. На индикаторе читаем 37,500009.

Тот факт, что значения синуса, вычисленные нами прежде и теперь, совпадают не полностью и что исходный угол получен обратно с маленькой «добавкой», обычен для вычислительной техники и не должен смущать.

Любой ЭВМ свойственно делать ничтожные ошибки. Это происходит из-за целого ряда причин; например, в нашем случае из-за того, что функции sin x и arcsin x подсчитываются по довольно сложным программам, заложенным в машину. Каждая из этих программ допускает ошибку в вычислениях, нередко достигающую шестого знака после запятой. Но заметьте: мы четыре раза ис-

пользовали эти программы, да еще два раза делали перевод из одной меры в другую, что влечет за собой умножение и деление. Остается только удивляться, что после таких «приключений» исходное число возвратилось назад с завидной точностью.

Процесс работы микрокалькулятора, вычислявшего sin и arcsin, был хорошо заметен: в это время на индикаторе около секунды ничего не было. Если бы мы взялись вручную повторить все то, что делала наша миниатюрная вычислительная машина, по таким же формулам и с такой же точностью, нам потребовалось бы не менее рабочего дня!

ОПЕРАЦИИ НАД ЧИСЛАМИ

Наш калькулятор, как и любая другая вычислительная машина, оперирует с числами. Важно понять, что при выполнении любого действия используемые числа не могут находиться в калькуляторе иным образом, как только записанными в память.

Числа запоминаются машиной в отведенных для этого ячейках, регистрах памяти.

Каждый регистр памяти в калькуляторе имеет свое обозначение в виде цифры или буквы. Десять из них обозначаются начальными натуральными числами от 0 до 9 включительно, еще четыре — буквами A, B, C, Д и еще пять — буквами X, Y, Z, T, X1. Группа последних регистров существенно отличается от остальных, и об этом мы поговорим подробнее на следующих страницах.

В дальнейшем для краткости мы будем обозначать регистры сокращенно: РХ, Р4 или РА.

При вводе в калькулятор число заносится в регистр X. От всех прочих он отличается тем, что его содержимое (т. е. записанное в него число) видно на индикаторе.

Все операции, при помощи которых вычисляются функции от некоторого числа, выполняются в нашем калькуляторе таким образом, что в качестве аргумента берется число из РХ и туда же помещается результат. Так было и с $\sqrt{}$5: после нажатия клавиш 5, F и $\sqrt{}$ значение корня было занесено в РХ, т. е. появилось на индикаторе.

Каждая такая операция (извлечение квадратного корня, возведение числа в квадрат, получение обратной величины от числа; вычисление синуса, косинуса, тангенса и обратных к ним функций; возведение в степень чи-

сел 10 и e; вычисление логарифмов, десятичного и натурального) выполняется над одним числом, и потому все они называются одноместными. Сложнее структура у арифметических операций — сложения и вычитания, умножения и деления. Все они двуместные, каждая выполняется над двумя числами. Порядок построения команд и в этом случае обратный: сначала калькулятору сообщают оба числа, а потом символ операции, которую требуется над ними совершить.

Каким бы странным ни казался такой порядок, у него есть определенные преимущества по сравнению с привычной для нас записью арифметических операций. Записывая их, мы не можем обойтись без скобок. Уберите их, например, в выражении $3{,}5\times(2{,}5-1)$, и оно превратится в $3{,}5\times2{,}5-1$, что приведет к другому результату. А команды для нашего калькулятора, как мы увидим чуть позже, в подобных случаях можно отдавать, не прибегая к скобкам.

Используемая в калькуляторных выкладках бесскобочная запись называется также польской, потому что впервые ее предложил польский ученый А. Лукасевич.

Числа, над которыми нужно совершить ту или иную арифметическую операцию, должны находиться в двух регистрах — РХ и РУ (оттого их и называют операционными). Ход в первый из них нам уже знаком: вводимое в него число набирается на клавиатуре. В РУ можно попасть только из РХ. Делается это нажатием клавиши ↑. При этом копия переданного числа остается в РХ. Затем туда записывается второе число; бывшее там прежде автоматически стирается.

Порядок расположения обоих чисел в регистрах РХ и РУ неважен, если их предстоит сложить или перемножить. В случае вычитания уменьшаемое должно находиться в РУ, вычитаемое — в РХ. В случае деления в РУ должно располагаться делимое, в РХ — делитель.

Введя числа в оба регистра, можно нажать клавишу выбранной операции. Результат ее будет помещен в РХ. То, что было прежде в РУ, не сохранится.

Ради примера вычислим арифметические выражения, встретившиеся нам пятью абзацами выше. Итак, $3{,}5\times(2{,}5-1)$. Начнем с выражения в скобках. Набираем на клавиатуре уменьшаемое 2,5 и нажимаем клавишу ↑. Число на индикаторе «мигнуло»: теперь оно переслано в РУ, а его копия осталась в РХ. Вводим вычитаемое (единицу) и нажимаем клавишу «—». Результат вычи-

тания читаем на индикаторе: 1,5. Казалось бы, для выполнения умножения его нужно переслать в РУ при помощи все той же клавиши ↑. Эту операцию, однако, можно сэкономить благодаря интересному свойству нашего калькулятора. Оказывается, если число на индикаторе является результатом некоторой операции, то оно передвигается в РУ, когда в РХ вводится новое число.

Набираем на клавиатуре 3,5. Теперь все готово для умножения: в РУ находится 1,5 и в РХ записано 3,5. Нажимаем клавишу со знаком умножения и читаем на индикаторе окончательный результат: 5,25.

Выполним вторую выкладку: 3,5×2,5−1. Продумайте последовательность нажатия клавиш и сравните ее с приведенной здесь:

Совершите все эти действия и сверьте результат с истинным: 7,75.

Теперь, когда мы познакомились со всеми вычислительными действиями, на которые способен наш микрокалькулятор, нужно отметить, что любое из них выполнимо лишь при соблюдении определенных условий.

Некоторые из них вытекают из самого определения математических операций, теряющих смысл при нарушении этих условий. Нельзя делить на нуль, извлекать квадратный корень из отрицательного числа, вычислять арксинус и арккосинус от числа, большего единицы, брать логарифм от неположительного числа, вычислять тангенс от числа вида $\pi/2 \pm \pi n$, возводить отрицательное число в произвольную степень. И если отдать микрокалькулятору команду о выполнении подобной некорректной операции, он остановится и на индикаторе появится надпись ЕГГОГ (подобного английскому error — ошибка).

Если результат какой-либо вычислительной операции превосходит $9{,}9999999 \cdot 10^{99}$, он не может быть показан на индикаторе, где умещаются лишь восемь знаков мантиссы и два знака порядка. Калькулятор и в этом случае выдает сообщение ЕГГОГ.

Последним обстоятельством, в частности, ограничены диапазоны аргументов, над которыми совершаются те или иные одноместные операции, выполнимые нашим

калькулятором. Эти диапазоны указаны в инструкции, прилагаемой к нему. Там же сказано, с какой погрешностью выполняется каждая из этих операций.

Вдумаемся в только что отмечавшуюся особенность нашего калькулятора: если в РХ ввести очередное число, то результат предыдущей операции перейдет в РУ. Вслед за этим можно выполнить действие с введенным в РХ числом и предыдущим результатом, находящимся в РУ. Получившийся результат опять перейдет в РУ при наборе нового числа и так далее. Такие действия называются цепочечными.

В качестве иллюстрации рассмотрим суммирование нескольких чисел: 11, 12, 13... Набираем число 11 и нажатием клавиши ↑ переводим его в РУ; вводим 12, нажимаем клавишу «+» и читаем на индикаторе результат: 23. Набираем число 13. При этом предыдущий результат 23 сразу переходит в РУ, и при нажатии клавиши «+» читаем сумму: 36. Теперь вводим 14... и так далее.

Возьмем формулу посложнее:

$$P=\sqrt{(a+b)\times c/d+e-f}.$$

Советуем и на сей раз самостоятельно предложить порядок действий и уже затем проверить себя по приведенному здесь:

Вычислите значение P при таких исходных данных: $a=10$, $b=11$, $c=12$, $d=-156$, $e=15{,}2$ и $f=-348{,}3$. Проверьте, получилось ли P=19,023265.

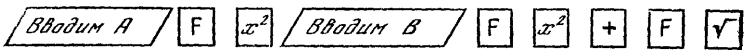
На результаты промежуточных вычислений можно не обращать особого внимания, ограничившись лишь быстрым контролем. Например, если выполнялось деление положительного и отрицательного чисел и частное получилось положительным, значит, при вводе не был уста-

новлен знак числа. Напомним: он устанавливается после того, как введены все цифры числа.

Неверно введенное число можно ввести заново: стираем его нажатием клавиши Cx, набираем его вновь и продолжаем работу.

В отличие от результата арифметического действия результат одноместной операции не изменяет содержимого РУ. Если над этим результатом, находящимся в РХ, опять провести одноместную операцию, то содержимое РУ опять не испортится. И так будет продолжаться до задания арифметической, двуместной операции.

Этот факт существенно облегчает проведение расчета по сложным формулам. Пусть, например, нам требуется вычислить гипотенузу прямоугольного треугольника C, у которого известны катеты A и B. Воспользуемся известной из геометрии теоремой Пифагора $C=\sqrt{A^2+B^2}$. Вычисления на калькуляторе по этой формуле сводятся к короткой цепочке действий:

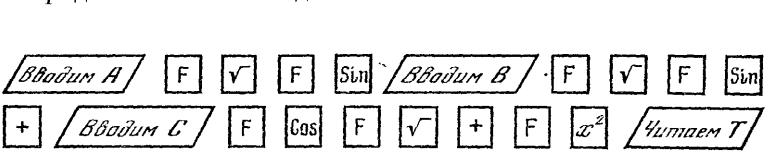


Проверьте, что при $A=3$ и $B=4$ получается $C=5$.

Теперь немного тригонометрии. Вот формула:

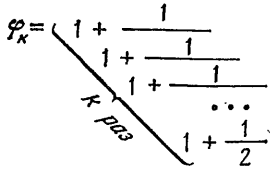
$$T=(\sin\sqrt{A}+\sin\sqrt{B}+\sqrt{\cos C})^2.$$

Порядок вычислений здесь может быть таким:



Для заключительной иллюстрации обратимся к пропорции золотого сечения, сыгравшей большую роль в истории математики и архитектуры. Выражающее ее число φ равно $(1+\sqrt{5})/2$. Вычислите его на калькуляторе. Должно получиться 1,6180339.

Известно несколько формул, позволяющих вычислить эту величину приближенно, но со сколь угодно высокой точностью — например:



Вычислите значение φ_k при $k=4$, 5, 6 и сравните его с величиной φ. Небольшая подсказка: начинайте вычисления с конца формулы.

Повторите эту процедуру еще и еще раз, удлиняя дробь. Как видите, результаты все менее отличаются от φ.

Вглядимся в нашу формулу повдумчивее. Знаменатель входящей в нее дроби — это по сути такая же формула, но укороченная на одно звено. Таким образом, наши действия можно описать так: от очередного приближения мы берем обратную величину и к результату прибавляем единицу — так получается следующее приближение. Далее процесс повторяется вновь. Самое начальное приближение — это двойка, стоящая в знаменателе последней дроби (недаром мы советовали начинать вычисления с конца!).

Этот вычислительный процесс можно выразить более короткой формулой $\varphi_{k+1}=1+1/\varphi_k$.

Нетрудно подметить сходство этого равенства с другим, которому подчиняется величина золотого сечения:

$$\varphi=1+1/\varphi.$$

Можно сказать, что предыдущее равенство получено из этого в такой форме, чтобы построить на его основе процесс последовательных приближений. Если употребить уже знакомый нам термин, можно отметить, что мы находим величину φ методом итераций.

Но это еще не все. В наших действиях можно усмотреть и более глубокий смысл. Они служат неплохим примером того, как решаются задачи с помощью вычислительных машин (будь то мощная ЭВМ или крохотный микрокалькулятор).

О РЕШЕНИИ ЗАДАЧ НА ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

Решение подразделяется на несколько этапов.

Первый этап — постановка задачи. Например, если говорить по-прежнему о золотом сечении, требуется по-

делить отрезок на две неравные части, чтобы большая относилась к меньшей так, как весь отрезок к большей. Именно в такой форме ставили задачу о золотом сечении древнегреческие математики, занявшиеся ею впервые.

Следующий этап — математическая формулировка задачи. Существенные для ее решения стороны изучаемого явления или объекта выражаются в виде количественных соотношений. Получившиеся при этом равенства, уравнения, неравенства образуют математическую модель исследуемого объекта или явления. В нашей задаче о делении отрезка, обозначив большую его часть A, меньшую — B, а их отношение — φ, легко получить:

$$A/B=(A+B)/A, \quad A/B=1+B/A, \quad \varphi=1+1/\varphi.$$

Это и есть математическая модель отрезка, поделенного в пропорции золотого сечения.

На следующем этапе выбирается метод решения. Метод должен быть выполнимым, должен гарантировать заданную точность решения и обеспечивать его получение за возможно меньшее время.

Поясним эти требования на нашем примере. Если нам нужно получить величину золотого сечения всего с двумя знаками после запятой, то выведенное нами уравнение для величины φ лучше свести к квадратному, решить его, получив $\varphi=(1+\sqrt{5})/2$, потом заглянуть в таблицу квадратных корней или на худой конец найти $\sqrt{5}$ подбором, выполнить сложение, деление — и дело с концом. Если требуется большая точность, можно получить значение корня на микрокалькуляторе, на нем же выполнить сложение и деление.

Ну а если имеющийся у вас микрокалькулятор настолько прост, что не умеет вычислять квадратные корни, а заданная точность по-прежнему высока?

Вот тут-то и пригодится уже знакомый нам метод итераций:

$$\varphi_{k+1}=1+1/\varphi_k.$$

Вслед за выбором метода решения наступает следующий этап: выбор алгоритма, т. е. последовательности конкретных действий, реализующих метод. Алгоритм для нахождения золотого сечения методом итераций мы уже описывали. Чтобы, пользуясь этим алгоритмом, получить искомую величину хотя бы с четырьмя верными знаками после запятой, требуется ни много ни мало... одиннадцать итераций. И возникает мысль: нельзя ли поручить

калькулятору не только выполнение, но и организацию этой работы, предписав ему, какие именно действия и в какой последовательности он должен выполнять?

Такое предписание называется программой. Программа для ЭВМ состоит из отдельных команд, выполнимых машиной. Программирование — один из дальнейших этапов решения задач на ЭВМ.

КОМАНДЫ МИКРОКАЛЬКУЛЯТОРА

Мы переходим к рассмотрению системы команд и приемов программирования на калькуляторе «Электроника БЗ-34».

Разговор об этом по-прежнему будем строить на конкретных примерах. Разумеется, на сей раз они будут посложнее, под стать теме разговора. Чтобы они усваивались легче, постараемся придать им развлекательный, игровой характер.

Перед вами — десять столбиков одинаковой толщины, длины которых (выраженные, скажем, в сантиметрах) нарастают в арифметической прогрессии, от 1 до 10. Что будет, если склеить их в торец и получившийся шест разрезать на десять равных частей? Какой будет длина «среднего» столбика? Решение этой задачи, по-видимому, не составит для вас труда: длина такого столбика равна 5,5.

Тогда — задача посложнее. Пусть те же числа от 1 до 10 на сей раз выражают диаметры кругов — скажем, блинов одинаковой толщины, которые приготовилась печь хозяйка. Что будет, если она слепит их все в один ком и сделает из него десять одинаковых блинов той же толщины? Каков будет диаметр «среднего» блина?

Тот, кто сообразит, что на сей раз среднее будет лежать между 6 и 7, продемонстрирует неплохую математическую смекалку. Но уже для того, чтобы предугадать, к какому из этих двух чисел будет ближе результат, требуется незаурядное чувство числа.

Впрочем, зачем полагаться на смекалку и на чувства, если в руках у нас программируемый микрокалькулятор? Составим для него программу для вычисления средней площади нескольких кругов и проверим свои прогнозы.

Для начала вспомним нужную формулу и вычислим по ней на микрокалькуляторе площадь какого-то одного из наших кругов.

Формула для площади круга хорошо известна: $S = \pi D^2/4$. Необходимые для расчета числа π и 4 имеются в калькуляторе, и их можно будет вызвать по ходу вычислений. Величину D надо ввести в регистр X заранее. Пусть, например, она равна 3.

Нажата клавиша 3, и то же число — на индикаторе, а значит, и в регистре X.

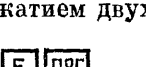
Если вести расчет по формуле вручную, то для этого, очевидно, далее надо нажать такие клавиши:

На индикаторе читаем результат: 7,0685832.

Те же клавиши и в той же последовательности нужно будет нажать, когда мы станем вводить в калькулятор программу для вычисления площади круга. В таком естественном порядке, воспроизводящем ход расчетов вручную, в калькулятор вводится любая программа для прямых вычислений по формулам, включающим выполнимые на нем математические операции. В этом заключается значительное удобство.

Программа не может располагаться в калькуляторе иначе, как в виде отдельных команд, каждая из которых занимает свою ячейку программной памяти (а некоторые команды — даже две ячейки). Всего таких ячеек 98. Им присвоены номера, называемые адресами,— от 00 до 97.

Чтобы ввести программу в калькулятор, надо перевести его в состояние, называемое режимом программирования. Делается это нажатием двух клавиш



Тотчас же в правом углу индикатора загорится 00. Это значит, что счетчик адресов установлен на нулевую отметку и команда, которую мы сейчас введем, займет адрес 00.

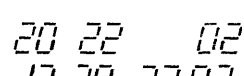
Ввод каждой новой команды станет увеличивать на единицу содержимое счетчика адресов, отображаемое в правом углу индикатора.

Нажимаем клавишу F. На индикаторе ничего не изменилось. Нажимаем клавишу x^2. В левом углу загорается 22. Это код операции возведения в квадрат. Его появление на индикаторе означает, что команда занесе-

на в программную память. Одновременно сменилось число в правом углу: сейчас там горит 01. По такому адресу разместится следующая введенная нами команда:

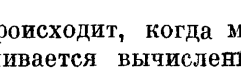
Нажимаем еще раз клавишу F, и вновь это не вызывает никаких перемен на индикаторе: клавиша F самостоятельного значения не имеет и образует команду лишь тогда, когда вслед за ней нажата еще одна клавиша.

Нажимаем клавишу л. Код 22 сдвинулся вправо, а на его месте появилось число 20 — код засылки числа л в регистр X. В правом углу — 02. По этому адресу разместится следующая команда, «умножить». Нажав соответствующую клавишу, замечаем: в левом углу оба кода разом сместились вправо, а на освободившемся месте загорелся код операции умножения 12;



Три кода выстроились на индикаторе от левого его края направо. Ввод каждой новой команды теперь будет приводить к тому, что левый и средний из этой тройки кодов сместятся на одну позицию вправо, стирая правый, а на освобождаемом ими месте появится код только что введенной операции. В правом же углу при этом загорится адрес, который будет занят командой, введенной вслед.

Вот как это происходит, когда мы вводим команды, по которым заканчивается вычисление площади круга:



В ячейки памяти калькулятора программа записывается как последовательность кодов операций.

Когда калькулятор выполнит первую команду введенной в него программы, содержимое счетчика адресов автоматически увеличится на единицу и калькулятор приступит к выполнению второй команды, выполнив ее

перейдет к следующей... По исполнении команды, дающей окончательный результат вычислений, машину нужно остановить. Для этого в программе ставится команда С/П. Нажав на клавишу с таким обозначением, завершаем ввод нашей программы. Текст ее запишем, указывая перед каждой командой ее адрес с точкой между ними:

00.Fx^2 01.Fπ 02.× 03.4 04.: 05.С/П

У клавиши С/П два назначения. Одно из них мы уже выяснили: если нажать ее в режиме программирования, то в программе появится команда останова. Чтобы познакомиться с другим назначением клавиши, вернем калькулятор в состояние, в котором он был сразу после включения.

Это состояние мы будем называть режимом вычислений. Находясь в нем, калькулятор либо выполняет отдельные команды, отдаваемые ему нажатием клавиш (режим ручных вычислений), либо автоматически вычисляет по имеющейся в его памяти программе (автоматический режим). Запуск калькулятора на автоматический счет и производится клавишей С/П. Отсюда ее обозначение, расшифровываемое как «Стоп/Пуск».

Чтобы вернуть калькулятор в режим вычислений из режима программирования, надо нажать две клавиши

F АВТ

Калькулятор готов к работе по программе. Но прежде чем запускать его, надо еще сообщить ему, с какой команды должен он начать вычисления.

Начальная команда введенной нами программы располагается по адресу 00. Калькулятор снабжен клавишей В/О, по сигналу которой счетчик адресов в режиме ручных вычислений возвращается на нулевую отметку — как говорят программисты, управление передается на адрес 00. Отсюда и обозначение В/О («Возврат на 0»). (Это не единственное назначение клавиши В/О — в свое время мы познакомимся и с другими.)

Итак, установим счетчик адресов на отметку 00. Введем в регистр X диаметр круга (пусть по-прежнему D=3). Запускаем калькулятор клавишей С/П, и он за несколько секунд выполнит одну за другой все команды

нашей крохотной программы. На индикаторе — площадь круга с диаметром 3, уже известная нам: 7,0685832.

Подсчитав на калькуляторе площадь круга вручную и по программе, иной читатель может разочарованно подумать, что никаких выгод по сравнению с ручным счетом программирование нам не принесло. Чтобы ввести программу, потребовалось нажать не только те же восемь клавиш, что и при работе вручную, но еще и семь «лишних»: F и ПРГ, чтобы перейти в режим программирования; С/П, чтобы калькулятор знал, где остановиться; F и АВТ, чтобы вернуться в режим вычислений; В/О и С/П, чтобы запустить программу на счет. 15 нажатий против 8. Конечно, такое сопоставление не в пользу программирования, если речь идет о единичном расчете по заданной формуле. Но ведь нам еще предстоит вычислять среднюю площадь кругов. Представьте, что их не десять, а двадцать, с целочисленными диаметрами от 1 до 20. Сколько раз придется нажать на клавиши, чтобы вычислить площади стольких кругов?

При ручном счете — 171 раз (можете проверить). А при счете по программе — всего 83 раза. В самом деле, 15 нажатий уйдет на то, чтобы получить первый результат. Но для получения каждого следующего нужно лишь после очередного останова вводить новое значение диаметра (одно нажатие, если число однозначное, два — если двузначное) и запускать программу на новый счет клавишами В/О и С/П. Программа, записанная в память калькулятора, сохраняется неизменной, покуда он не выключен. Для нового использования ее не нужно вводить заново.

83 против 171 — в таком сопоставлении выигрыш от программных расчетов налицо.

Он заметен тем отчетливее, чем больше программа и чем чаще приходится проводить по ней повторяющиеся вычисления. Так оно бывает, например, при составлении таблиц. А методы последовательных приближений, столь характерные для вычислительной математики? В них каждое приближение выполняется одной и той же последовательностью команд. И наконец, игры. Каждый ход — это новый счет по игровой программе. А увлекательная игра, конечно, не заканчивается за два-три хода, как шахматная задача.

Большие программы отличаются еще и тем, что в вычислениях по ним, как правило, используется много исходных данных (а не одно-единственное число, как в

нашем учебном примере с площадью круга). Хранят их в так называемых адресуемых регистрах памяти, обозначаемых цифрами (от 0 до 9) и буквами (А, В, С, Д). Нетрудно подсчитать, что всего их 14.

Числа направляются на хранение в эти регистры командами засылки. Перед выполнением такой команды засылаемое в память число сначала должно оказаться в регистре X (в результате набора на клавиатуре, вызова из памяти или выполнения какой-либо операции), и уже оттуда оно направляется в адресуемый регистр. Извлечь число из адресуемого регистра можно командой вызова; при этом оно направляется опять-таки в регистр X.

Команда засылки выполняется с помощью клавиши П (от слова «Память»), команда вызова — с помощью клавиши ИП («Из Памяти»).

Освоим обе команды сначала в ручном исполнении. Наберем на клавиатуре какое-нибудь число, например 3. Оно тотчас появляется на индикаторе и, стало быть, находится сейчас в регистре X.

Допустим, мы хотим поместить его в регистр 4. Нажимаем клавишу П и затем — клавишу 4. Число на индикаторе мигнуло: теперь оно и в регистре X, и в регистре 4. То, что было раньше в регистре 4, пропадает.

Очистим регистр X, нажав клавишу Сх. На индикаторе — нуль. Затем нажимаем клавиши ИП и 4. На индикаторе — снова 3: число переписано из регистра 4. Но и там оно не пропало: в этом можно убедиться, вторично очистив индикатор и нажав клавиши ИП 4.

Опробуем теперь те же операции в программном варианте. Нет, речь пока пойдет не о том, что составленная нами программа для вычисления площади круга войдет составной частью в программу для определения средней площади нескольких кругов. Предположим, что в какой-то другой большой программе по ходу расчета потребовалось вычислить площадь круга. Его диаметр берется из регистра 4 (напомним, что мы уже поместили туда число три), а найденная площадь засылается в регистр 5. Допустим, что цепочка команд, выполняющих этот расчет, должна размещаться в программе, начиная с адреса 60.

Устанавливаем счетчик адресов на нужный адрес в режиме ручных вычислений с помощью клавиши БП («Безусловный Переход»). Нажимаем сначала ее, а потом набираем на клавиатуре 60. Это число будет гореть в правом углу индикатора, когда нажатием клавиш F

ПРГ мы переведем калькулятор в режим программирования. Именно по 60-му адресу запишется команда, которую введем мы первой.

Нажимаем клавишу ИП. На индикаторе — никаких перемен. Нажимаем клавишу 4. В левом углу загорается 64. Это код операции вызова в регистр X содержимого регистра 4. То, что он загорелся лишь после нажатия второй клавиши, свидетельствует: клавиша ИП самостоятельного значения не имеет и образует команду вызова лишь в сочетании с другой клавишей, обозначающей номер регистра, содержимое которого вызывается в регистр X.

Вводим далее команды, по которым определяется площадь круга. Затем нажимаем клавиши П и 5: ими в программу вводится команда засылки результата в регистр 5. Наблюдая в это время за индикатором, можно убедиться, что клавиша П также не имеет самостоятельного значения и образует команду засылки в сочетании с нажатой далее клавишей, цифровой или буквенной.

Завершим ввод нажатием клавиши С/П. Вот текст введенной нами программы:

60.ИП4 61.Fx^2 62.Fп 63.× 64.4 65.: 66.П5 67.С/П.

Клавишами F АВТ возвращаем калькулятор в режим вычислений. Клавишами БП 6 0 устанавливаем счетчик адресов на 60-й адрес. Клавишей С/П запускаем программу. Через несколько мгновений вычисления заканчиваются и на индикаторе загорается знакомое нам число 7,0685832. Нажав клавиши ИП 5, можно убедиться, что оно размещается в регистре 5. А заодно и в том, что в момент останова по команде С/П в регистре X находится результат выполнения предыдущей команды.

Вот теперь мы и перейдем к составлению программы для подсчета средней площади нескольких кругов, куда составной частью войдет прежняя наша программа для вычисления площади круга.

Вычислив ее при очередном значении диаметра, следует прибавить результат к сумме площадей; попутно надо подсчитывать количество кругов, а когда они все будут перебраны, полученную сумму нужно будет поделить на это количество.

Обе эти величины можно шаг за шагом накапливать в адресуемых регистрах: скажем, количество кругов — в нулевом, сумму их площадей — в первом. Получив по

составленной нами программе площадь очередного круга в регистре X, можно вызвать туда из регистра 1 накопленную на этот момент сумму площадей. Содержимое регистра X при этом сдвинется в регистр У. Затем остается выполнить сложение — так к сумме площадей прибавится новое слагаемое. Результат сложения, как мы знаем, получится в регистре X. Оттуда его нужно отослать в регистр 1.

Подобным же образом, вызвав содержимое регистра 0, затем единицу и сложив оба числа, получим пополнившееся количество кругов и отошлем его обратно в регистр 0 — регистр-счетчик.

К этой цепочке команд надо приписать еще команду останова и команду возврата на нулевой адрес. Когда счет остановится на первой из них, надо ввести в калькулятор диаметр нового круга и нажать клавишу С/П. Программа перейдет к следующей команде, совершит по ней возврат на адрес 00 и проведет расчет для нового круга.

Чтобы рассчитать среднее арифметическое, надо приписать к нашей программе еще четыре команды: вызов содержимого регистра 1 (сумма площадей), вызов содержимого регистра-счетчика 0 (количество кругов), деление (оно даст интересующую нас среднюю площадь) и останов (на индикаторе будет светиться результат предыдущей операции, т. е. ответ на поставленную задачу). На первую из этих команд мы передадим управление с помощью клавиши БП, когда переберем все круги, и после останова считаем искомый ответ с индикатора.

Это описание нетрудно перевести на язык команд. Но прежде надо решить, как организовать возврат на нулевой адрес. Дело в том, что команда В/О передает управление на адрес 00 только в режиме ручных вычислений. Стало быть, вместо нее в программе придется использовать команду безусловного перехода — ту самую, которую мы уже исполняли недавно с помощью клавиши БП.

У этой команды — своя тонкость. Когда мы станем вводить ее в программу (забегая вперед, скажем, что счетчик адресов к этому моменту будет показывать 16) и нажмем клавишу БП, содержимое счетчика адресов тотчас же увеличится на единицу: 17. Когда же мы наберем адрес перехода, содержимое счетчика адресов увеличится еще раз: 18. Это означает, что команда БП — «двойная», т. е. занимает в программе два адреса. В пер-

вом из них записывается операция БП, во втором — адрес перехода.

Отказавшись от команды В/О, мы свободны в выборе этого адреса. Учтем, что перед запуском программы следует очистить регистры 0 и 1, предназначенные для накопления сумм. Очистку можно произвести вручную, нажав клавиши Сх П 0 П 1. Но лучше, чтобы это сделал калькулятор, и притом с самого начала: 00.Сх 01.П0 02.П1 03.С/П.

Команда останова здесь не лишняя, да и в дальнейшем пригодится: на нее-то мы и станем передавать управление, вычислив площадь очередного круга, и, покуда калькулятор бездействует, будем вводить диаметр следующего круга.

С учетом высказанных соображений программа примет такой вид:

00.Сх 01.П0 02.П1 03.С/П 04.Fx^2 05.Fп 06 ×
07.4 08.: 09.ИП1 10.+ 11.П1 12.ИП0 13.1
14.+ 15.П0 16.БП 17.03 18.ИП1 19.ИП0 20.:
21.С/П.

Программа получилась объемистая — будем внимательны при ее вводе. Введя, нажмем клавиши В/О С/П. Очистив за три первые команды регистры 0 и 1, калькулятор остановится. Введем диаметр первого круга (1), запустим программу, после останова введем диаметр второго круга (2), запустим программу вновь... и так будем продолжать, пока не будет вычислена площадь последнего круга (D=10). Передадим управление на адрес 18 и вскоре после нового пуска получим на индикаторе искомую среднюю площадь кругов: 30,237829.

С помощью нашей карманной ЭВМ нетрудно выяснить, что такую площадь имеет круг с диаметром, равным 6,2048368. А какую величину назвали вы, когда пытались угадать это число?

СТЕК И ПРИНЦИПЫ ЕГО РАБОТЫ

Тот, кто знакомится с программированием на предлагаемых нами примерах, вряд ли представит себе, что ответ на поставленную нами задачу можно получить по более короткой программе. Между тем программисты не зря говорят: всякую программу можно сократить по крайней мере на одну команду. Сейчас мы докажем это на примере нашей задачи. Правда, для этого нам придется освоить несколько новых команд.

Ранее, поговорив об одноместных операциях и перейдя к рассмотрению двуместных, мы сразу отметили, что для их выполнения необходимы два операционных регистра: X и Y. Мы узнали, как с помощью клавиши ↑ в них вводятся участники арифметических действий и где размещается результат.

Ради простоты мы умолчали тогда, что перемещение чисел охватывает при этом не только регистры X и Y, но также Z и T. Эти четыре регистра находятся в тесной взаимосвязи и объединяются названием «стек» (английское слово, означающее «охапка, стопка»). К ним примыкает регистр X1, куда после выполнения многих операций направляется прежнее содержимое регистра X.

То, что мы узнали прежде, дополним здесь до завершенной картины всевозможных перемещений чисел по стеку.

Посмотрим сначала, что происходит в нем, когда в регистр X засылается новое число. Пусть оно вызывается туда из какого-то адресуемого регистра (командами ИП1, ИПА и т. п.) или из ячейки, где хранится число π (командой Fπ). Тогда прежнее содержимое PX смещается в PY, содержимое PY — PZ, содержимое PZ — в PT, прежнее содержимое регистра T пропадает. И если представить регистры стека один над другим (X внизу, T вверху — см. рис. 10, 11), то описанное перемещение удобно назвать движением снизу вверх. Содержимое регистра X1 при этом не меняется.

Когда новое число вводится в регистр X с клавиатуры, числа в стеке тоже перемещаются снизу вверх, за исключением случаев, когда перед этим с клавиатуры вводилось какое-то число и вслед за его набором были выполнены команды ↑ или Cx. Тогда при вводе нового числа в PX с остальными регистрами стека и с регистром PX1 ничего не происходит.

Если после набора числа на клавиатуре не нажать клавишу ↑, то при дальнейшем нажатии цифровых клавиш все будет происходить так, как если бы мы продолжали вводить цифры предыдущего числа.

Кстати сказать, аналогичным образом можно ввести число в регистр X и в режиме автоматических вычислений по программе. Например, цепочка команд 00.3 01., 02.1 03.4 запишет в регистр X число 3,14.

По команде ↑ числа, находившиеся в стеке, сдвигаются по его регистрам снизу вверх, причем в PX остается копия числа, находившегося там прежде.

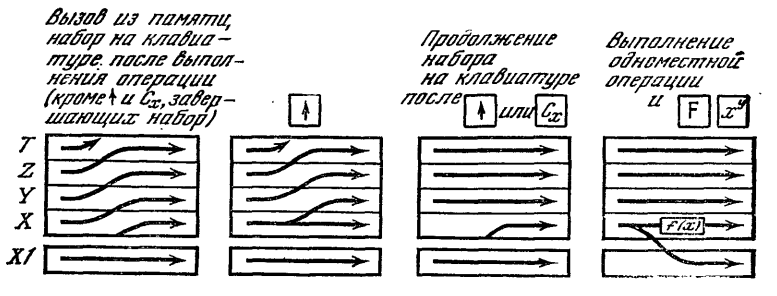


Рис. 10

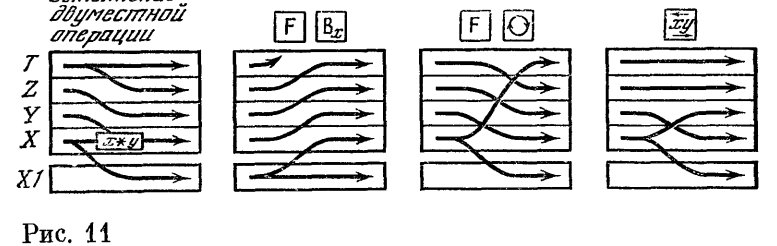


Рис. 11

По команде Cx содержимое регистра X стирается (точнее, заменяется нулем), а в прочих регистрах стека и в регистре X1 все остается по-прежнему.

Когда над числом, находящимся в регистре X, совершается какая-либо одноместная операция, в регистрах Y, Z, T также ничего не происходит, а в регистр X1 отправляется прежнее содержимое регистра X. Таким же образом выполняется и операция x^y.

Каждая арифметическая операция выполняется так: ее участники берутся из регистров X и Y, результат направляется в регистр X. Прежнее содержимое регистра X отсылается в регистр X1, прежнее содержимое регистра Y утрачивается — туда спускается содержимое PZ. Содержимое PT, в свою очередь, смещается в PZ, оставляя на прежнем месте свою копию.

При засылке числа из регистра X в любой адресуемый регистр сохраняется прежнее содержимое и регистра X, и всех других регистров стека, и регистра X1.

Вот еще три команды, по которым движутся числа по регистрам стека.

Команда FBx вызывает содержимое регистра X1 в регистр X. Числа в регистрах стека при этом сдвигаются

снизу вверх. В регистре X1 остается копия находившегося там прежде числа.

По команде ⇄ содержимое PY переходит в PX, содержимое PX — в PY и PX1. С ее помощью удобно выводить на индикатор содержимое регистра Y.

Команда F○ осуществляет круговое движение чисел по стеку: из PY — в PX, из PZ — в PY, из PT — в PZ, из PX — в PT и PX1. С ее помощью можно просмотреть на индикаторе содержимое всех регистров стека. Эта команда оказывается полезной и тогда, когда число, введенное в регистр X для какой-то промежуточной операции, сдвинет вверх прежнее содержимое PX, PY и PZ. Восстановить прежнее размещение чисел по этим регистрам можно, выполнив команду F○.

Научившись управлять движением чисел по стеку, мы сможем сократить последнюю из составленных нами программ на целых четыре команды:

00.Cx 01.↑ 02.С/П 03.Fx^2 04.Fл 05.× 06.4
07.: 0.8+ 0.9⇄ 10.1 11.+ 12.⇄ 13.БП 14.02
15.⇄ 16.: 17.С/П.

Здесь начальные команды Cx и ↑ очищают регистры X и Y перед последующим накоплением количества кругов и суммы их площадей. После останова в регистр X вводится диаметр первого круга, калькулятор запускается вновь, после останова вводится диаметр следующего круга и снова нажимается клавиша С/П...

К моменту очередного останова в регистре X находится сумма площадей кругов с 1-го по некоторый k-й (обозначим ее Σ_k), в регистре Y — их количество k. В этом нетрудно убедиться с помощью таблицы, в последовательных столбцах которой указаны адреса, команды, коды операций, а также содержимое регистров стека после выполнения каждой команды (рис. 12).

Первые два столбца заполним по тексту программы. Ручные операции ввода (когда в PX заносится диаметр нового круга) и последующего пуска программы клавишей С/П запишем в таблицу без адресов.

Проходя по строчкам таблицы сверху вниз, заполняйте остальные столбцы в соответствии с результатами выполнения команд, и при вычислении площади $(k+1)$-го круга она примет вид, подтверждающий, что программа работает так, как мы предполагали (рис. 12; прочерк означает, что содержимое регистра сохранилось неизменным).

Попробуйте продолжить эту таблицу до завершающего адреса 17, проверьте, что к моменту останова по этому адресу в регистре X (и на индикаторе) находится искомое среднее арифметическое.

Адрес	Команда	Код операции	Содержимое регистров				
			X	Y	Z	T	X1
02	с/п	50	Σ_k	k			k
Ввод D_{k+1}			D_{k+1}	Σ_k	k		—
Пуск			—	—	—		—
03	Fx^2	22	D^2_{k+1}	—	—		D_{k+1}
04	$F\pi$	20	π	D^2_{k+1}	Σ_k	k	—
05	×	12	πD^2_{k+1}	Σ_k	k	—	π
06	4	04	4	πD^2_{k+1}	Σ_k	k	—
07	:	13	$\pi D^2_{k+1}/4$	Σ_k	k	—	4
08	+	10	Σ_{k+1}	k	—	—	$\pi D^2_{k+1}/4$
09	$\leftrightarrows$	14	k	Σ_{k+1}	—	—	Σ_{k+1}
10	1	01	1	k	Σ_{k+1}	—	Σ_{k+1}
11	+	10	$k+1$	Σ_{k+1}	k	—	1
12	$\leftrightarrows$	14	Σ_{k+1}	$k+1$	—	—	$k+1$
13	БП	51	—	—	—	—	—
14	02	02	—	—	—	—	—

Рис. 12

Введите программу в калькулятор, проведите расчет с теми же исходными данными, что прежде, и убедитесь, что результат получается тот же: 30,237829.

Заметим, что подобные таблицы очень облегчают разбор программ — и своих, и особенно составленных другими.

ЦИКЛЫ. КОСВЕННАЯ АДРЕСАЦИЯ

Попытаемся теперь переделать нашу последнюю программу так, чтобы уменьшить количество ручных операций. Ведь чем их больше, тем выше риск ошибки.

Программа распадается на две части. В первой (адреса 00—14) вычисляется площадь очередного круга, подсчитывается сумма всех до сих пор перебранных кругов и их количество. Во второй части (адреса 15—17) вычисляется среднее арифметическое площадей. По какой цепочке команд пойти, определяем мы сами, в первом случае просто запуская программу после очередного останова, во втором — совершая безусловный переход на адрес 15.

Нельзя ли поручить выбор пути самому калькулятору?

Вопросы такого рода могут возникнуть при составлении многих программ, даже весьма несложных. Например, при решении квадратного уравнения: если его дискриминант положительный или равен нулю, то корни отыскиваются по одним формулам, если отрицательный — то по другим. Или возьмем пример из тригонометрии: при положительном аргументе арккосинус связан с арксинусом одним соотношением, при отрицательном — другим. И если по ходу работы программы может возникнуть любая из двух каких-то альтернативных возможностей, программа должна содержать фрагменты для расчета каждого варианта и работать по тому или другому из них в зависимости от выполнения или невыполнения некоторого условия. Фрагменты такого характера называют ветвями.

На эти случаи в нашем калькуляторе предусмотрены команды условного перехода. Они вводятся в программу с помощью клавиш, над которыми написаны соотношения: $x<0$, $x=0$, $x\geqslant 0$, $x\neq 0$. Судя по этим обозначениям, проверка любого альтернативного условия должна быть сведена к сравнению с нулем содержимого регистра X.

В силу «надклавишного» расположения надписей ввод команд условного перехода в программу предваряется нажатием клавиши F. Нажав вслед за ней клавишу, помеченную выбранным соотношением (скажем, $x\neq 0$), замечаем, что счетчик адресов увеличился на единицу. Он вновь увеличится на единицу, когда затем мы введем адрес перехода. Как видим, каждая команда условного перехода — двойная, занимает в программе два адреса. Например: 03.F$x\neq 0$ 04.17.

На указанный в такой команде адрес перехода управление передается, если условие не выполняется. Так в нашем примере, если содержимое регистра X равно нулю и неравенство $x\neq 0$ не соблюдается, совершается переход на адрес 17. Если же условие выполняется, т. е. $x\neq 0$, управление передается на адрес, следующий за командой условного перехода, в нашем примере — на адрес 05.

Для иллюстрации подобных переходов мы не случайно привели фрагмент 03.F$x\neq 0$ 04.17. Оказывается, если вставить его в нашу программу, она будет гораздо удобнее.

00.Cx 01.↑ 02.С/П 03.F$x\neq 0$ 04.17 05.Fx^2 06.Fπ
07.× 08.4 09.: 10.+ 11.⇄ 12.1 13.+ 14.⇄
15.БП 16.02 17.F○ 18.⇄ 19.: 20.С/П

Покуда мы будем вводить в калькулятор один за другим диаметры кругов, выраженные, разумеется, ненулевыми числами, команда условного перехода будет передавать управление на адрес 05, начальный адрес первой части программы, где вычисляются и суммируются площади кругов, подсчитывается их количество. А когда расчет этих величин закончится, введем нуль. Поскольку условие $x \neq 0$ теперь не выполнено, программа перейдет на адрес 17, начальный адрес второй части, где будет вычислена средняя площадь кругов.

По адресу 17 в программу, как видно, вставлена новая команда: F○. Необходимость в ней возникает потому, что нуль, введенный в регистр X для перехода на заключительную часть программы, нарушил нужное нам расположение величин в стековых регистрах: суммарная площадь кругов сместилась из РХ в РУ, количество кругов — из РУ в РZ. Команда F○ возвращает эти величины на свои места.

Теперь для перехода к заключительной стадии расчета надо нажимать всего две клавиши, 0 и С/П, вместо четырех: БП, двух цифр адреса перехода, С/П. Не нужно вспоминать или сверяться по инструкции к программе, каков этот адрес. Заботы об этом калькулятор берет на себя.

Вычисление средней площади кругов для нас всего лишь игра. Но вполне может статься, что подобный расчет придется выполнить по ходу решения какой-то крупной задачи. И тогда в программу для ее решения цепочка команд, вычисляющих среднюю площадь, войдет в качестве фрагмента.

Допустим, он должен разместиться в большой программе начиная с 60-го адреса. Предположим, что к началу его работы диаметры кругов уже находятся в адресуемых регистрах: D_1 — в РД; D_2 — в РС, D_3 — в РВ, D_4 — в РА, D_5 — в Р9, D_6 — в Р8 и т. д. (Такой «обратный» порядок обнаружит свой смысл позже.) Предположим еще, что длина массива заполненных регистров известна и не превышает 12, так что регистр 1 остается свободным, а выражающее эту длину число кругов находится в регистре 0. (Эти облегчающие предположения не поблажка читателю, они естественны для программ, составленных опытным программистом, который всегда думает об удобствах своей работы.)

Подсчет средней площади кругов теперь представляется совсем простым: вызывать один за другим диамет-

ры кругов из последовательных регистров, вычислять по ним и суммировать площади кругов, а затем поделить сумму на число слагаемых. Но для такой процедуры нужны соответствующие команды: команда условного перехода, по которой он совершается заданное число раз; команда вызова, которая при многократном ее исполнении вызывает в регистр X числа из последовательных адресуемых регистров.

Нужные нам команды условного перехода вводятся в программу с помощью клавиш F L0, F L1, F L2, F L3. Будем объединять эти пары клавиш обозначением FLM. Как и при вводе всякой команды условного перехода, нажав такую пару, мы должны далее ввести адрес перехода. В программе такая команда займет два адреса: в первом — операция FLM, во втором — две цифры, означающие адрес перехода.

Допустим, он меньше того, под которым записана операция FLM, а в регистре M находится целое число N, большее единицы. Выполнив цепочку команд, предшествующих операции FLM, программа приступает к выполнению команды условного перехода. Операцией FLM из содержимого регистра M вычитается единица, и полученная разность сравнивается с нулем. Если она не равна нулю, то засылается в регистр M, а управление передается по адресу перехода. Цепочка тех же команд выполняется еще раз, снова из содержимого регистра M вычитается единица... Так продолжается до тех пор, пока результат вычитания не окажется равным нулю. В таком случае он не засылается в регистр M и там остается единица; управление же передается на команду, следующую за командой условного перехода, т. е. через адрес от операции FLM.

Это происходит, как нетрудно установить, после N-кратного прохождения цепочки. Засылая в регистр M нужное число, мы можем задавать количество таких прохождений.

Фрагмент программы, выполняемой много раз подряд, называется циклом. Поэтому операция FLM, образующая только что описанную команду условного перехода, именуется операцией организации цикла.

Есть среди команд нашего калькулятора и нужная нам команда вызова чисел из последовательных регистров. В программу она вводится последовательным нажатием трех клавиш: К, ИП и той, что указывает номер одного из адресуемых регистров. Условно обозначим этот

номер М и предположим, что в нем находится целое число.

Несмотря на обилие клавиш, требуемых для ее ввода, эта команда занимает в программе один адрес и записывается слитно: КИПМ. Работает она по-разному в зависимости от значения М.

Пусть М равно одному из чисел от 0 до 3 включительно. Тогда по команде КИПМ содержимое регистра М уменьшается на единицу и в РХ вызывается содержимое того регистра, номер которого равен получившейся разности.

Пусть М равно одному из чисел от 4 до 6 включительно. Тогда по команде КИПМ содержимое регистра М увеличивается на единицу и в РХ вызывается содержимое того регистра, номер которого равен получившейся сумме.

Пусть М представляет собой число от 7 до 9 или букву от А до Д. Тогда по команде КИПМ в РХ вызывается содержимое того регистра, номер которого равен числу, находящемуся в регистре М.

Это число может превышать 9. Скажем, если оно равно 10, то калькулятор поймет его как обозначение регистра А, 11 — как В, 12 — как С, 13 — как Д. Именно 13 получится, например, в регистре 1 при первом выполнении команды КИП1, если до того там находилось 14.

Команды вида КИПМ называются командами косвенного вызова. Есть у нашего калькулятора и сходные с ними по структуре команды косвенной засылки. Набираются они нажатием клавиши К, П и еще одной, указывающей номер некоторого регистра М. Каждая занимает в программе один адрес.

Работают они аналогично командам косвенного вызова и тоже в предположении, что в регистре М находится целое число. Вот, скажем, что происходит по команде КП1: из содержимого регистра 1 вычитается единица и в регистр, номер которого равен получившейся разности, направляется содержимое регистра Х.

Кстати, приведем соответствующий термин: уменьшение или увеличение содержимого каких-либо регистров при использовании команд косвенного вызова и косвенной засылки называется модификацией адреса.

Располагая командами косвенного вызова и организации циклов, нетрудно составить задуманный нами фрагмент:

60.ИП0 61.1 62.4 63.П1 64.Сх 65.КИП1 66.Fx^2

67.Fπ 68.× 69.4 70.: 71.+ 72.FL0 73.65
74.⇄ 75.:

Начертите таблицу, с помощью которой можно следить за движением чисел по стеку, и понаблюдайте по ней, как «работает» только что составленная нами цепочка команд.

Команда ИП0 вызывает из регистра 0 в регистр X количество кругов; оно понадобится в самом конце для нахождения среднего арифметического. Четыре дальнейшие команды засылают в регистр 1 число 14, используемое в работе следующей команды КИП1, и очищают регистр X: в нем далее будет накапливаться сумма площадей. При первом выполнении команды КИП1 из начального содержимого регистра 1, т. е. из числа 14, вычтется единица и получится 13 — номер регистра Д. Из него-то и будет вызвано значение первого диаметра. Цикл команд, записанных под адресами 65—71, в первый раз вычислит площадь первого круга и командой сложения начнет подсчет суммы площадей; во второй раз он вычислит уже площадь второго круга; ведь при втором исполнении команды КИП1 из содержимого ячейки 1 вновь вычтется единица и получится 12, номер регистра С. Там находится значение второго диаметра — его-то и вызовет на сей раз команда КИП1 в регистр X. Команда FL0 65 (адреса 72—73) будет вновь и вновь передавать управление на начало цикла, и каждый раз из содержимого регистра 0, куда вначале было записано количество кругов, будет вычитаться по единице. Сколько кругов, столько раз и будет пройден цикл. По выходе из него в регистре X — сумма площадей кругов, в регистре Y — их количество. Их надо вначале переставить (адрес 74), чтобы затем поделить (адрес 75) и получить тем самым искомую среднюю площадь.

В разобранном нами фрагменте программы нет команды останова. Можно предположить, что полученный результат будет незамедлительно использован в следующей части программы. Но возможно и другое толкование: расчет, производимый этим фрагментом, требуется проводить в различных местах «большой» программы. Писать его в тексте программы несколько раз неэкономно. Лучше каждый раз передавать на него управление и возвращаться обратно.

Для организации таких переходов с возвратом служит клавиша ПП («Переход к Подпрограмме»). Вслед

за ее нажатием нужно ввести в программу начальный адрес фрагмента, на который нужно передать управление. Такой фрагмент и называется подпрограммой. Команда перехода к нему, как нетрудно понять из нашего описания, занимает два адреса. На следующий за ними адрес произойдет возврат, если в конце подпрограммы поставить команду В/О.

Судя по сказанному до сих пор, система команд у «Электроники БЗ-34» весьма богатая: мы сумели выполнить все свои замыслы при составлении и совершенствовании задуманных программ. Некоторые возможности нашего микрокалькулятора при этом даже остались неиспользованными и неописанными.

Между тем на будущее стоило бы рассказать про некоторые варианты и аналоги описанных выше команд.

Например, нажатием соответствующих клавиш можно набрать команду КП↑ или КИП↑. Работают они совершенно аналогично командам КП0 и КИП0, отличаясь лишь тем, что не изменяют содержимого регистра 0.

(Кстати, эти две команды действуют иначе на микрокалькуляторах «Электроника МК-61» и «Электроника МК-52». Между тем на них выполнимы все остальные команды «Электроники БЗ-34», а сверх этого — еще и целый ряд других команд. Это означает, если употребить строгий термин, что система команд моделей «МК-61» и «МК-52» расширяет систему команд модели «БЗ-34». Поэтому по программам, написанным для «БЗ-34», можно вести расчеты на «МК-61» и «МК-52», если, повторяем, в этих программах нет команд КП↑, КИП↑ и других, набор которых завершается нажатием на клавишу ↑.)

Упомянем здесь же команды условного перехода Кx<0М, Кx=0М, Кx≥0М, Кx≠0М, где число в регистре М указывает адрес перехода, а также аналогичные им команды безусловного перехода КБПМ и перехода к подпрограмме КППМ. Каждая из них занимает всего один адрес, а не два, как уже знакомые нам команды перехода, однако требует для своей работы один адресуемый регистр.

(Кстати, введем новый термин: всякий раз, когда адрес засылки, вызова или перехода указывается не прямо, а со ссылкой на ячейку памяти, где он хранится, говорят о косвенной адресации.)

Если в регистре М находится дробное число, большее единицы, то при выполнении команды КИПМ там остается лишь целая часть числа. При выполнении команд

КИП0, КИП1, КИП2, КИП3 это получившееся целое число еще уменьшается на единицу, а при выполнении команд КИП4, КИП5, КИП6 на единицу увеличивается.

ЕЩЕ О РЕШЕНИИ ЗАДАЧ НА ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

Нам не пришлось рассматривать все богатство возможностей «Электроники БЗ-34» потому, что до сих пор мы решали довольно простые задачи. Будь они посложнее, мы не принимались бы сразу после их постановки за составление программ, а более последовательно проходили бы этапы их решения, предшествующие счету по программе.

Несколькими страницами ранее, прослеживая эти этапы на примере задачи о золотом сечении, мы остановились на одном из первых, когда строится алгоритм решения. Поговорим сейчас об этом этапе подробнее.

Здесь надо прежде всего изыскивать возможности для сокращения вычислительной работы, в частности, упрощать расчетные формулы. Такая возможность упрощения есть, например, в только что рассматривавшейся нами задаче о средней площади кругов с диаметрами от D_1 до $D_к$ (будем продолжать наш разговор на этом примере). Подсчитывая сумму их площадей, лучше было бы не вычислять каждую по формуле, а суммировать лишь квадраты диаметров и потом умножить на $\pi/4$ всю полученную сумму.

Продумывая алгоритм, желательно выделить в расчетных формулах повторяющиеся комбинации переменных и принять эти комбинации за промежуточные переменные. Их целесообразно вычислить однажды и потом использовать в готовом виде. Затем следует установить порядок выполнения операций, а попутно наметить, как разместить по регистрам (адресуемым и стековым) исходные данные, промежуточные переменные и окончательные результаты.

Если бы, наверстывая упущенное, мы составили сейчас подобную роспись регистров для программы о вычислении средней площади кругов, у нас получилось бы так: диаметры кругов $D_к$ — в адресуемых регистрах начиная с 13-го и далее в порядке убывания номеров (притом в таком количестве n, чтобы оставались незанятыми регистры 0 и 1); регистры 0 и 1 — счетчики (обозначим счетный индекс k, регистр 0 используем для орга-

низации цикла, регистр 1 — для косвенного вызова диаметра $D_к$); число кругов направляем в регистр Y, начальное (нулевое) значение суммы квадратов их диаметров в регистр X (далее обе величины будут бок о бок путешествовать по регистрам стека, а когда все круги будут перебраны, мы вернем их на прежние места).

Алгоритм решения задачи с учетом недавней поправки сейчас представляется таким: прибавлять один за другим квадраты диаметров $D_к^2$ к их сумме $\sum_n = D_1^2 + D_2^2 + \ldots + D_к^2$ и повторять этот процесс циклически n раз, затем разделить сумму на n и умножить на $\pi/4$.

Коротко и ясно. К сожалению, для более сложных задач словесное описание алгоритма их решения получается далеко не столь коротким и оттого плохо обозримым. Поэтому, прежде чем приниматься за составление программы, алгоритм изображают в графической форме.

Каждая стадия решения описывается при этом словами и формулами, заключенными в замкнутую рамку. Такие фигуры называются блоками. Они соединяются линиями, указывающими последовательность стадий. Вся конструкция именуется блок-схемой алгоритма. Составлять ее следует так, чтобы по каждому блоку можно было написать достаточно независимые и обозримые куски программы.

Составление блок-схемы алгоритма — это следующий этап решения задачи на ЭВМ.

Всякая такая схема состоит из блоков четырех различных очертаний: овалов, параллелограммов, прямоугольников и ромбов. Форма блока соответствует характеру действий, в нем описываемых. В параллелограммах обозначаются ввод и вывод информации, в прямоугольниках — конкретные вычисления (если они выполняются в подпрограмме, то вертикальные стороны прямоугольника чертятся двойными), в ромбах — проверки условий (условные переходы). В овалах пишутся слова «начало» и «конец»; такие фигуры располагаются по концам блок-схемы.

Блоки обычно располагаются сверху вниз и часто нумеруются. Соединительные линии рисуются в виде прямолинейных отрезков или ломаных с вертикальными и горизонтальными звеньями, иногда снабжаются стрелками. Если же на соединительных линиях стрелки отсутствуют, то предполагается, что по вертикальным прямым движение совершается сверху вниз, а по горизонтальным — слева направо. Из каждого ромба выходят две ли-

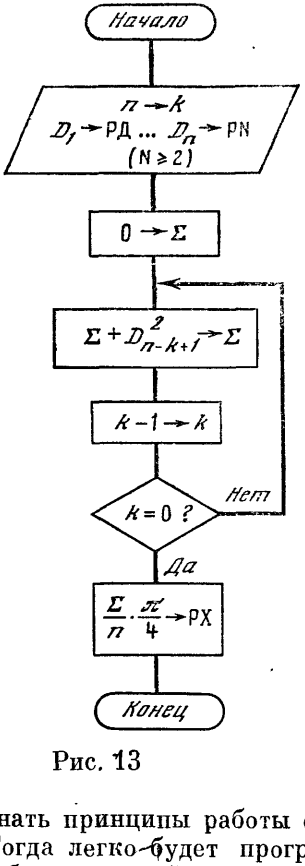


Рис. 13

нии. Одна соответствует выполнению условия и помечается словом «да», вторая невыполнению и помечается словом «нет».

Составим блок-схему алгоритма для подсчета средней площади кругов (рис. 13). Для завершенности картины будем на сей раз рассматривать задачу как самостоятельную.

Начинающему рекомендуем нарисовать блок-схему в более подробном варианте (рис. 14). Этот вариант таков, что в его блоках уже видятся конкретные команды. Например, блок L—1→L PL→PX явно выразится в программе командой КИПМ, где M=0, 1, 2 или 3. Блоки k—1→k k=0? выразятся операцией FLM, где M — одно из тех же чисел, но иное, нежели выбранное для команды КИПМ.

К сожалению, в блок-схеме трудно отразить движение информации по стеку: это приходится домысливать. Вот, в частности, почему следует хорошо знать принципы работы стека и уметь ими пользоваться. Тогда легко будет программировать вычисления по алгебраическим формулам, составляющим вычислительные блоки.

Следующий этап решения задач на ЭВМ — программирование. Об этой работе говорилось уже достаточно, так что читатель, вероятно, сможет без подсказок написать новую программу для вычислений средней площади кругов, где суммируются лишь квадраты их диаметров. Свой вариант мы напишем здесь не так, как раньше, не в строчку, а столбцами: адрес, команда, код операции (рис. 15).

Записанную так программу легче читать, легче вводить в калькулятор и отлаживать, о чем мы будем вскоре говорить.

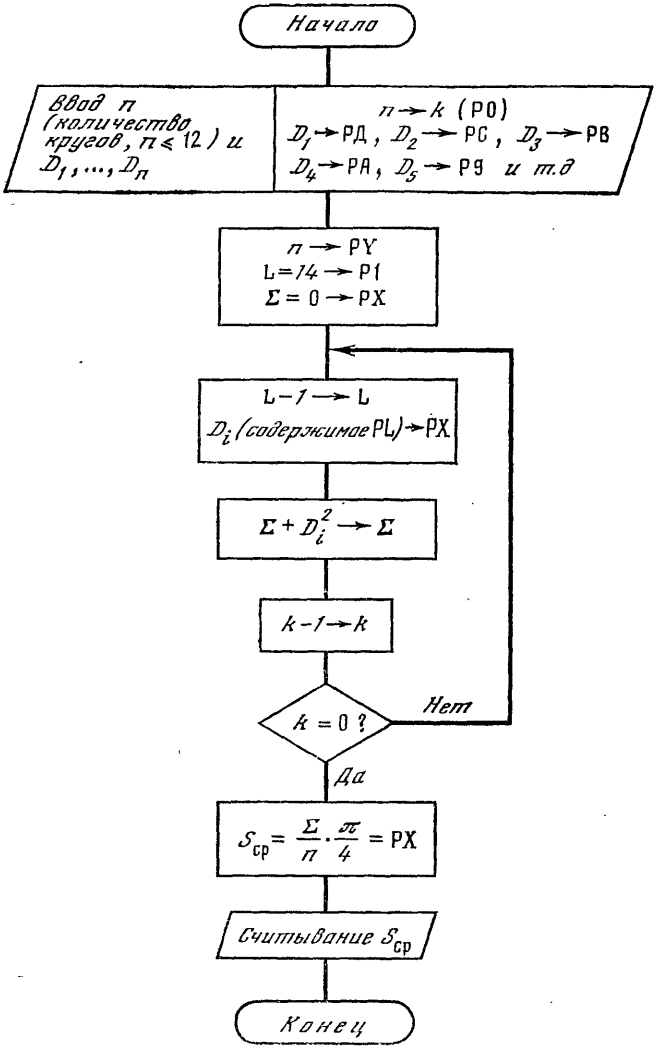


Рис. 14

Длина у этой программы такая же, как и у прежней, да и все команды те же. Но умножение на коэффициент $\pi/4$ вынесено за пределы цикла и выполняется не при каждом его прохождении, а однократно. Меньше умножений — короче срок получения результата. Как приняго

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ИП0	60	06	Fx^2	22	I2	FП	20
0I	I	0I	07	+	I0	I3	×	I2
02	4	04	08	FLO	5Г	I4	4	04
03	ПI	4I	09	05	05	I5	:	I3
04	Cx	0Г	I0	≠	I4	I6	С/П	50
05	КИПI	ГI	II	−:	I3			

Рис. 15

говорить, мы оптимизировали программу по времени ее выполнения.

Впрочем, мы уже выходим за пределы этапа программирования. Оптимизация программы — составная часть ее отладки. Так называется этап, начинающийся со ввода программы в калькулятор и заключающийся в проверке, действительно ли она выполняет алгоритм, который должна выполнять.

Сначала стоит проверить, верно ли введена программа. Возвращаемся в режим вычислений, передаем управление на начальный адрес программы и вновь переводим калькулятор в режим программирования. Затем раз за разом нажимаем клавишу ШГ со стрелкой вправо («ШГ вправо», как мы условимся ее называть). С каждым ее нажатием содержимое счетчика адресов, отображаемое в правой стороне индикатора, увеличивается на единицу, так что у левого края индикатора последовательно появляются коды, записанные в программную память. Адрес крайнего левого кода всегда на единицу меньше того, что в данный момент виден у правого края.

Также можно просмотреть всю программную память. Если после очередного нажима клавиши «ШГ вперед» слева на индикаторе появился не тот код, что записан в тексте программы, следует вернуться на адрес назад, нажав клавишу «ШГ назад», а потом заново набрать команду, которая должна стоять по адресу, высвечиваемому справа. Прежнее содержимое этого адреса сотрется и заменится кодом набранной операции.

«Двойные» команды переходов следует набирать заново целиком: и операцию перехода, и адрес перехода.

Если какая-то команда оказалась ненужной, на ее

место следует вставить команду, набираемую на клавишах К НОП («Нет Операции»). Никаких действий калькулятора она не вызывает, а адрес в программе занимает, так что остальные команды остаются при своих адресах и программу не приходится переписывать и вводить заново.

Так можно добиться полного совпадения текстов — введенного в память калькулятора и записанного на бумаге.

Окончательно убедившись, что программа правильно введена в калькулятор, проверим, правильно ли она работает.

Если она невелика, то проще всего сделать это так. В режиме ручных вычислений передаем управление на начальный адрес, а затем раз за разом нажимаем клавишу ПП («Потактовый Проход» — так раскрывается это обозначение, если вычисления выполняются вручную). С каждым ее нажатием последовательно выполняется по одной команде программы, и результат выводится на индикатор. Особенно легко протекает такая проверка, если составить контрольный пример, для которого можно сказать, какие результаты должны получиться после выполнения каждой команды.

Может случиться, что после неудачной проверки программу придется составлять заново. Когда же программа отлажена окончательно, т. е. верность выдаваемых ею результатов гарантируется, можно приняться за ее оптимизацию — попытаться сократить время ее работы, сделать ее более удобной для пользователя и т. п.

Готовую программу следует очень аккуратно оформить. Ведь достаточно одной неразборчиво написанной команды или кода — и пользоваться ею будет невозможно. Оформление программы — следующий этап решения задачи.

Для этой цели лучше всего взять лист плотной бумаги. В заголовке помещают название программы — несколько слов, кратко и точно поясняющих ее суть. Желательно указать, как используются по ходу счета адресуемые регистры, сколько времени уходит на получение каждого результата. Далее следует собственно текст программы: адреса, команды, коды по столбцам. Далее — инструкция по пользованию программой. По пунктам, коротко и понятно описывается работа человека от ввода текста программы, начальных данных и констант до получения результатов. Далее — контрольный пример с конкретными значениями начальных данных и получающихся резуль-

ОПРЕДЕЛЕНИЕ СРЕДНЕЙ ПЛОЩАДИ

кругов с диаметрами $D_1 \ldots D_n$ $(n \leq 12)$

$$S_{ср} = \frac{\left(\sum_{i=1}^{n} D_i^2\right)}{n} \cdot \frac{\pi}{4}$$

В регистр 0 перед началом счёта заносится количество кругов n, в регистры Д,С,В,А,9,...2 – диаметры кругов $D_1 \ldots D_n$.

Регистры 0,1 – счётчики. Результат $S_{ср}$ – на индикаторе.

Время счёта 20 с.

ПРОГРАММА

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ИП0	60	06	$F x^2$	22	12	$F\pi$	20
01	1	01	07	+	10	13	×	12
02	4	04	08	FL0	5Г	14	4	04
03	ПI	41	09	05	05	15	:	13
04	Сх	0Г	10	≠	14	16	С/П	50
05	КИПI	ГI	11	:	13			

ИНСТРУКЦИЯ

Пункт	Действия	Нажимаемые клавиши	На индикаторе
1	Переход в режим программирования	F ПРГ	
2	Ввод программы	по тексту программы	
3	Переход в режим вычислений	F АВТ В/0	
4	Ввод количества кругов n и их диаметров $D_1 \ldots D_n$	n П0 D_1 ПД D_2 ПС D_3 ПВ D_4 ПА D_5 П9 и так далее до окончания массива	n D_1 D_2 D_3 D_4 D_5 .... D_n
5	Вычисление средней площади кругов $S_{ср}$	С/П	$S_{ср}$

Контрольный пример:

$n=10$, $D_1=1$, $D_2=2$, $D_3=3, \ldots D_{10}=10$ $\quad S_{ср}=30,23783$

Рис. 16

татов. Как бы тщательно ни вводилась программа, лишь решение контрольного примера дает уверенность, что она введена без ошибок.

Вот как может выглядеть лист с только что составленной нами программой для вычисления средней площади кругов (рис. 16).

Заключительный этап решения задачи на ЭВМ — работа с программой.

В завершение рассказа о приемах работы с программируемым микрокалькулятором «Электроника Б3-34» разберем несколько практических примеров. Первый из них потребует лишь ручных вычислений. Для решения следующих двух понадобится составить программы.

КАК ДАЛЕКО ДО ГОРИЗОНТА?

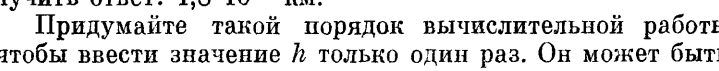
Еще в школе на уроках географии нам говорили: дальность горизонта составляет около 5 км. Выясним с помощью нашей карманной ЭВМ, верно ли это.

Дальность горизонта — это расстояние от смотрящего до самой далекой доступной взору точки на равнине. Поскольку поверхность земли выпуклая, взор не может простираться за горизонт.

Построив несложный чертеж, легко выразить дальность горизонта L через радиус Земли R и рост смотрящего h: $L=\sqrt{h(2R+h)}$.

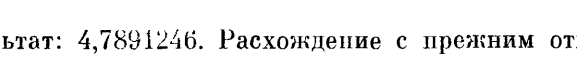
Кривизна земной поверхности неодинакова в различных ее местах. Среднее значение радиуса кривизны принимается равным $6371{,}032=6{,}371032\cdot10^3$ км (пристрастные к точности, выпишем все цифры, приводимые в справочнике). Величину человеческого роста, которую нужно подставить в формулу, положим равной 180 см. Выразим эту величину в километрах, чтобы в тех же единицах получить ответ: $1{,}8\cdot10^{-3}$ км.

Придумайте такой порядок вычислительной работы, чтобы ввести значение h только один раз. Он может быть, например, таким:



Складываем, умножаем, извлекаем корень... На индикаторе — 4,7891249. В самом деле, дальность горизонта — около пяти километров.

А теперь получим тот же результат по упрощенному алгоритму. В сомножителе $(2R+h)$ под корнем отбросим величину h.



Результат: 4,7891246. Расхождение с прежним ответом — только в седьмом знаке после запятой. Погрешность весьма мизерная, а алгоритм упростился значительно: мы сэкономили три нажатия на клавиши. В режиме ручных вычислений это не очень-то заметно, но когда речь идет о программе, то к подобному выигрышу, если он сокращает и длину программы, и время вычислений по ней, следует стремиться всегда. Такие достижения оттачивают мастерство программирования. Человек, не овладевший им, то и дело будет попадать в неприятные оказии, когда программа для решения стоящей перед ним задачи не умещается в памяти микрокалькулятора, а если и умещается, то счет по ней грозит затянуться на многие часы.

Бывает, конечно, что требования быстроты и точности вступают в противоречие. Тогда разумен такой подход к разрешению альтернативы: пусть программа будет подлиннее (лишь бы умещалась в памяти калькулятора!), но ответ дает за наименьшее время и, разумеется, с заданной точностью.

ПАРАДОКСЫ ОПИСАННЫХ МНОГОУГОЛЬНИКОВ

При бесконечном увеличении числа сторон многоугольника, описанного около некоторой окружности, его периметр стремится к длине этой окружности.

Возьмем этот факт за повод к тренировке в составлении несложных программ. Радиус окружности положим для простоты равным единице. Тогда периметр многоугольника будет стремиться к 2π=6,2831853. Исходный многоугольник пусть будет у нас правильным, а увеличивать число его сторон станем удвоением. Тогда сторона правильного $2n$-угольника a_{2n} выразится через сторону правильного n-угольника a_n формулой

$$a_{2n} = \frac{4}{a_n}\left(\sqrt{1+\frac{a_n^2}{4}}-1\right).$$

Алгоритм, по которому вычисляются периметры таких многоугольников, составить несложно. Сначала надо занести в адресуемые регистры, скажем в P0 и P1, соответственно число сторон исходного многоугольника n и длину его стороны a_n. Удобно начать с описанного квадрата, т. е. взять $n=4$; тогда при единичном радиусе окружности $a_4=2$. После ввода исходных данных калькулятор запускается на счет нажатием клавиши С/П. Первыми же командами программы определяется периметр многоугольника: число его сторон n умножается на их длину a_n. Калькулятор останавливается, чтобы можно было прочесть результат. Снова нажата клавиша С/П — и дальнейшими командами программы вычисляется a_{2n} и заносится в P1. Удваивается n и заносится в P0. Затем управление передается на ту команду, с которой начинается вычисление периметра. Вычислив его, калькулятор останавливается вновь, давая возможность прочесть новый результат, и все повторяется снова и снова.

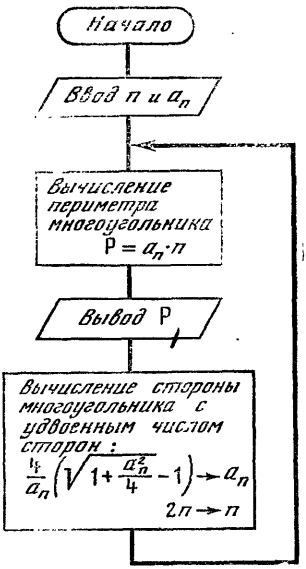


Рис. 17

Нарисуйте блок-схему алгоритма, составьте реализующую его программу и сравните с помещенной здесь (рис. 17).

Пояснения к программе (рис. 18):

00—02: величины a_n и n, введенные действиями $n\uparrow a_n$ в регистры X и Y соответственно, отсылаются в отведенные для них адресуемые регистры 1 и 0;

03—05: вычисляется и выводится на индикатор периметр многоугольника;

06—19: вычисляется и отсылается в регистр 1 длина стороны многоугольника с удвоившимся числом сторон;

20—23: число сторон многоугольника удваивается и отсылается в регистр 0;

24—25: управление передается на адрес, с которого начинается вычисление периметра многоугольника.

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ПI	4I	09	:	I3	I8	×	I2
0I	FO	25	I0	I	0I	I9	ПI	4I
02	П0	40	II	+	I0	20	ИП0	60
03	ИПI	6I	I2	F√	2I	2I	2	02
04	×	I2	I3	I	0I	22	×	I2
05	С/П	50	I4	−	II	23	П0	40
06	ИПI	6I	I5	ИПI	6I	24	БП	5I
07	F x^2	22	I6	:	I3	25	03	03
08	4	04	I7	4	04			

Рис. 18

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ПI	4I	09	I	0I	I8	ПI	4I
0I	FO	25	I0	+	I0	I9	С/П	50
02	П0	40	II	F√	2I	20	ИП0	60
03	ИПI	6I	I2	I	0I	2I	2	02
04	×	I2	I3	−	II	22	×	I2
05	ИПI	6I	I4	ИПI	6I	23	П0	40
06	F x^2	22	I5	:	I3	24	БП	5I
07	4	04	I6	4	04	25	03	03
08	:	I3	I7	×	I2			

Рис. 19

Приступаем к расчету. В помещенной ниже таблице в первом столбце — число сторон многоугольника, во втором — его периметр.

4	8,0000000	64	6,2881546
8	6,6274160	128	6,2840264
16	6,3651890	256	6,2824186
32	6,3034376		

Два верных знака после запятой уже есть. Нажимаем клавишу С/П вновь и вновь, надеясь на дальнейшее повышение точности.

Но что это? Точность, вопреки ожиданиям, становится все ниже, а потом на индикаторе, что уже совсем невероятно, загорается нуль!

512	6,2756812	4096	5,9773058
1024	6,2490016	8192	4,4909104
2048	6,1750018	16384	0

Разобраться в этих парадоксах нам поможет та же программа, если в ней поменять местами команду останова 05.С/П и команды, стоящие по адресам 06—19 (рис. 19).

Теперь в момент останова на индикаторе будет появляться не периметр, а длина стороны очередного многоугольника. Мы увидим тогда, что при числе его сторон, измеряемом тысячами, длина стороны выражается десятичной дробью с двумя-тремя нулями после запятой. Квадрат такой дроби, поделенный на четыре, как того требует формула для вычисления a_{2n}, есть дробь с пятью-шестью нулями после запятой. Например, $a^2_{4096}/4 = 0{,}00000053239145$.

Что будет, если такую величину прибавить к единице? На бумаге записать ответ нетрудно:

$$1 + a^2_{4096}/4 = 1{,}000000\ 53239145.$$

Но индикатор нашего микрокалькулятора содержит всего лишь восемь десятичных разрядов. Все, что не вмещается в это прокрустово ложе, отбрасывается, причем число в младшем из сохраняемых разрядов округляется. И получается:

$$1 + a^2_{4096}/4 = 1{,}0000005.$$

Формула для a_{2n} требует далее извлечь из этой суммы квадратный корень, а из полученного числа вычесть единицу. Проследим, каким должен быть точный результат и каким он получается на калькуляторе (в скобках):

$$\sqrt{1 + a^2_{4096}/4} = 1{,}00000026619569\ (1{,}0000002),$$

$$\sqrt{1 + a^2_{4096}/4} - 1 = 2{,}6619569 \cdot 10^{-7}\ (2 \cdot 10^{-7}).$$

Погрешность — более 30 процентов!

При дальнейшем нарастании числа сторон величина $a_n^2/4$ становится меньше 10^{-7} и при сложении с единицей напрочь выходит за пределы разрядной сетки индикатора. Все получается так, как если бы дробь $a_n^2/4$ равнялась нулю. Подставьте такое ее значение в числитель формулы для a_{2n}, и вы получите $a_{2n}=0$. Так же считает и микрокалькулятор.

Подобные казусы типичны для неумелого использования метода последовательных приближений.

Какова погрешность, с которой микрокалькулятор вычисляет очередной результат, выводимый на индикатор? Насколько очередное приближение отличается от истинного значения искомой величины? Как снизить погрешность? Как построить процесс последовательных приближений, чтобы он гарантировал достижение ответа на поставленную задачу? Чтобы ответить на эти вопросы, порой требуется провести глубокое математическое исследование, заменить метод решения, перестроить алгоритм...

(Несложный пример такой перестройки можно продемонстрировать и в нашей задаче об описанных многоугольниках. Путем тождественных преобразований формулу для a_{2n} можно видоизменить следующим образом:

$$a_{2n}=\frac{a_n}{\sqrt{1+(a_n/4)\,a_n}+1}.$$

Как видим, здесь устранен один из самых коварных источников погрешностей — вычитание близких величин. Некоторое повышение точности дает устроенное в знаменателе чередование операций умножения и деления. Насколько успешны предпринятые изменения, читатель может убедиться сам, проведя расчеты заново).

Но даже и не вдаваясь в глубины вычислительной математики, можно посоветовать всякому, кто занимается расчетами на микрокалькуляторе: умейте представлять себе, как распределяются по десятичным разрядам знаки чисел, с которыми по написанной вами программе производятся математические операции, как урезаются эти числа до размеров разрядной сетки индикатора, как округляются.

Внимание к каждому разряду, умение использовать каждый разряд очень помогают при составлении игровых программ.

Ну, а теперь — игра.

Составление игровых программ ставит свои проблемы. Если мы хотим, чтобы микрокалькулятор стал нашим партнером, нам надо позаботиться о том, чтобы он понимал наши сообщения, а мы — его.

Калькулятор способен высказываться только одним способом — выводя какие-то числа на индикатор. А на язык чисел не так-то просто бывает перевести правила даже простенькой игры.

Возьмем для примера уже знакомую нам игру «Угадай число». Ведущий называет двузначное число от 10 до 99 включительно. Пытаясь его отгадать, игрок называет пробные двузначные числа, и на каждое из них ведущий отвечает либо «многовато», либо «маловато», смотря по тому, больше или меньше оно, чем задуманное.

Если роль ведущего поручить микрокалькулятору, то он, конечно, не сможет вывести на индикатор слова «многовато» и «маловато». Но можно поступить так. Сначала на индикаторе появляется число 1099, напоминая игроку, что задуманное калькулятором двузначное число находится в промежутке от 10 до 99. Игрок вводит в калькулятор свое пробное число — обозначим его AB. Калькулятор вычитает его из задуманного. Тотчас устраивается проверка: не равна ли разность нулю? То есть не разгадано ли заданное число? Если разгадано, то надо сообщить игроку о его успехе подобающим образом. Традиционная школьная оценка за отличный результат — пятерка. Можно вывести на индикатор шеренгу пятерок: 55555555. Это число получается в результате операций $5 \cdot 10^8/9$.

Если же пробное число не совпадает с загаданным, надо еще выяснить, больше или меньше оно, чем загаданное, а для этого посмотреть на знак, уже полученной нами разности обоих чисел.

Предположим, она положительна. Стало быть, задуманное число лежит в промежутке между AB и 99. Пожалуй, понятнее всего будет сообщить об этом игроку так: вывести на индикатор число $AB99$. Если же разность между задуманным и пробным числом отрицательна, т. е. загаданное число находится между 10 и AB, то на индикатор надо вывести число $10AB$.

Число $AB99$ получается из AB просто: $AB \cdot 100 + 99 = AB99$. Число $10AB$ не сложнее: $10 \cdot 100 + AB = 10AB$.

Казалось бы, можно уже приниматься за составление программы. Осталась мелочь: придумать, как калькулятор станет задумывать число из диапазона между 10 и 99.

Ответ на этот «пустяковый» вопрос требует не меньше пояснений, чем все сказанное нами раньше по поводу нашей игры.

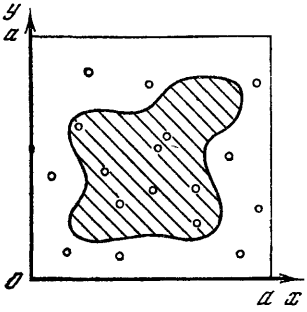


Рис. 20

Дело в том, что по самому ее смыслу загаданное число должно быть неожиданным для игрока, случайным. Слово, прозвучавшее в конце предыдущей фразы,— не эпитет, а чисто научный термин. Проблема получения случайных чисел важна не только при составлении игровых программ, но и с чисто научной, и с сугубо практической точек зрения. Так что проблема эта заслуживает весьма обстоятельного разговора.

Представьте себе, что вам требуется определить площадь фигуры, граница которой выражается довольно сложной функцией. Один из путей к ответу — вычислить интеграл от функции. Но задача интегрирования таит в себе немало математических и вычислительных трудностей. И если точность, с которой должен быть получен ответ, не так уж велика, можно пойти другим путем.

Фигура вычерчивается на координатной плоскости и заключается в рамку, площадь внутри которой найти совсем уж просто,— например, в квадрат, две стороны которого — оси координат, а две другие отстоят от осей на расстоянии a (рис. 20). Берется несколько точек, координаты которых представляют собой случайные числа в диапазоне $(0, a)$. Вероятность того, что такая случайная точка попадет внутрь фигуры, равна отношению площади фигуры к площади квадрата. С ростом числа точек частота попаданий все менее отличается от вероятности попадания. Отсюда вытекает простой прием: взять внутри квадрата побольше случайных точек, выяснить, какие из них попадают в границы фигуры (с математической точки зрения это гораздо легче, чем считать интеграл), и составить пропорцию, где слева стоит отношение числа попавших в эти границы точек к общему их числу, а справа — отношение площади фигуры к площади

квадрата. Решив столь несложную пропорцию, получаем площадь фигуры.

Такой метод называется методом Монте-Карло. Свое название он получил от небольшого средиземноморского городка, где находится знаменитый дом для игры в рулетку, и получил неспроста: выигрыш в этой азартной игре — дело чистого случая.

Метод Монте-Карло — не единственный, где находят применение случайные числа. Поэтому немалый интерес представляет разработка программ, позволяющих строить (или, как говорят специалисты, генерировать) последовательности таких чисел, лежащих в заданном диапазоне. К таким программам предъявляется целый ряд требований: каждое число последовательности должно получаться возможно быстрее, заданный диапазон должен заполняться этими числами равномерно и т. д. Самое же главное требование: генерируемые программой числа должны быть действительно случайными, такими, чтобы появление каждого из них нельзя было предугадать.

К сожалению, этому важнейшему требованию применяемые на практике программы не удовлетворяют. Выдаваемая любой из них числовая последовательность периодична, т. е. образуется повторением некоторой цепочки чисел. Эта цепочка может быть очень длинной, состоять из десятков и сотен миллионов чисел, но коль скоро уж есть периодичность, появление очередного числа нельзя назвать непредсказуемым. Потому-то такие числа и называются не случайными, а псевдослучайными.

Откуда берется каверзная периодичность, можно понять на примере двух генераторов псевдослучайных чисел, приводимых в книге А. Н. Цветкова и В. А. Епанечникова «Прикладные программы для микро-ЭВМ «Электроника БЗ-34», «МК-56», «МК-54». Последовательности таких чисел строятся по формулам:

а) 1-я программа: $\xi_{k+1}=\{11\xi_k+\pi\}$;

б) 2-я программа: $\xi_{k+1}=(1/\pi)\arccos(\sim\cos(10^9\xi_k))$.

Здесь фигурные скобки означают дробную часть заключенной в них величины, а символ $\sim$ напоминает, что от столь больших значений аргумента косинус вычисляется микрокалькулятором с высокой погрешностью.

Каждое из чисел, генерируемых той и другой программой, лежит в диапазоне $]0,1[$. Развернутые вовне квадратные скобки в подобной записи означают, что границы

промежутка, в данном случае нуль и единица, в него не входят. Следовательно, каждое число той и другой последовательностей имеет порядок -1 или еще меньший, а значащих цифр в его мантиссе восемь — больше не умещается на индикаторе. Для ясности будем представлять каждое такое число в привычном виде десятичной дроби: нуль, запятая и восемь цифр после запятой. Сколько существует таких дробей? Каждый знак после запятой может варьироваться от 0 до 9. Стало быть, всего вариантов по всем знакам 10^8. Если взять более чем 10^8 таких дробей, то две из них обязательно совпадут. Предположим теперь, что две такие дроби ξ_k и ξ_l $(k<l)$ встретились в ряду чисел, выдаваемых, скажем, первым генератором. Если они одинаковы, то одинаковыми будут и следующие за ними в ряду: $\xi_{k+1}=\{11\xi_k+\pi\}=\{11\xi_l+\pi\}=\xi_{l+1}$. Одинаковыми будут также ξ_{k+2} и ξ_{l+2}. Короче говоря, вся цепочка чисел от ξ_k до ξ_l повторится, начиная с ξ_l, а затем будет повторяться все снова и снова.

Длина повторяющегося фрагмента в последовательности псевдослучайных чисел, выдаваемых первой программой,— около 8000 чисел, второй — около 4500. Время получения очередного псевдослучайного числа — соответственно 3 и 4,5 с, длина программ — 12 и 9 адресов, диапазон между нулем и единицей заполняется генерируемыми числами весьма равномерно. Поэтому в расчетах на микрокалькуляторах обе программы находят широкое и успешное применение.

Если требуются псевдослучайные числа не из диапазона $]0,1[$, а из какого-то иного промежутка $]A,B[$, пересчет делают по формуле $\eta_k=A+(B-A)\xi_k$. Бывает и так, что требуются не произвольные, а лишь целые числа из некоторого диапазона. Так и в нашей игре: калькулятор должен «задумать» целое число от 10 до 99 включительно. Тогда над дробной величиной η_k нужно совершить еще дополнительные действия. Например, отбросить ее дробную часть. Если число $\eta_k>1$, то проще всего заслать его в регистр M с номером 7, 8, 9, A, B, C или D и выполнить затем команду КИПМ. Тогда в регистре M останется лишь целая часть числа η_k. Нетрудно сообразить, что образующиеся целые числа займут диапазон $[A,B-1]$ (включая граничные его точки).

Можно указать для той же цели другой прием. Сначала сложим η_k с 10^7. Тогда дробная часть η_k уйдет за рамки восьмиразрядной сетки индикатора, а младший из сохраненных разрядов целой части будет округлен. Затем

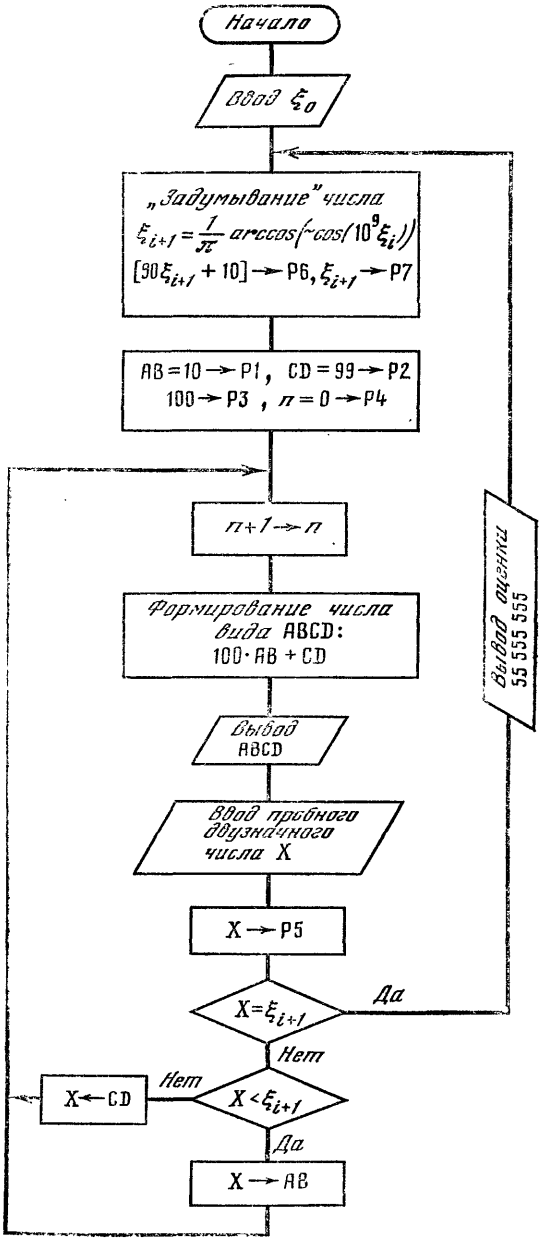

Начало
Ввод ξ_0
„Задумывание" числа
$\xi_{i+1} = \frac{1}{\pi} \arccos(-\cos(10^9 \xi_i))$
$[90\xi_{i+1} + 10] \rightarrow$ P6, $\xi_{i+1} \rightarrow$ P7
AB = 10 → P1, CD = 99 → P2
100 → P3, n = 0 → P4
n+1 → n
Формирование числа вида ABCD:
100·AB + CD
Вывод ABCD
Ввод пробного двузначного числа X
X → P5
$X = \xi_{i+1}$
Да
Вывод цифры 55 555 555
Нет
$X < \xi_{i+1}$
Нет
X ← CD
Да
X → AB

Рис. 21

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	П7	47	23	7	07	46	П5	45
01	ИП7	67	24	+	10	47	ИП6	66
02	ВП	0С	25	Г Вх	0	48	-	11
03	9	09	26	-	11	49	Г x=0	5Е
04	Fcos	1Г	27	П6	46	50	59	59
05	Farccos	1-	28	1	01	51	5	05
06	Fπ	20	29	0	00	52	ВП	0С
07	:	13	30	П1	41	53	8	08
08	П7	47	31	↑	0Е	54	↑	0Е
09	9	09	32	×	12	55	9	09
10	0	00	33	П3	43	56	:	13
11	×	12	34	9	09	57	С/П	50
12	1	01	35	9	09	58	В/О	52
13	0	00	36	П2	42	59	Г x<0	5С
14	+	10	37	Сх	0Г	60	65	65
15	↑	0Е	38	П4	44	61	ИП5	65
16	5	05	39	КИП4	Г4	62	П1	41
17	↑	0Е	40	ИП1	61	63	БП	51
18	9	09	41	ИП3	63	64	39	39
19	:	13	42	×	12	65	ИП5	65
20	-	11	43	ИП2	62	66	П2	42
21	1	01	44	+	10	67	БП	51
22	ВП	0С	45	С/П	50	68	39	39

Рис. 22

вычтем из суммы 10^7. Так будут получаться целые числа из диапазона $[A, B]$. Чтобы происходило не округление, а именно отбрасывание дробной части, надо еще вычитать из суммы η_k+10^7 число 0,55555555. Так станут возникать целые числа из уже знакомого нам «усеченного» диапазона $[A, B-1]$.

Последний метод мы и применим, составляя программу для игры «Угадай число». Примем $A=10$, $B=100$. Тогда числа, «загадываемые» микрокалькулятором, окажутся в нужном нам диапазоне от 10 до 99 включительно.

Собственно говоря, у нас уже все готово для составления алгоритма и написания программы. Осталось лишь перечислить все используемые для вычислений константы и назначить адресуемые регистры для их хранения. Например, 10 поместим в Р1, 99 — в Р2, 100 — в Р3. Набирать эти числа удобно командами программы. Вводимое игроком пробное число пусть хранится в Р5, задуманное калькулятором — в Р6, а псевдослучайное число из диапазона]0,1[, используемое для «загадывания»,— в Р7. Будем вести счет попыткам игрока угадать хранимое в памяти калькулятора число. Пусть регистром-счетчиком будет Р4. Прибавлять к его содержимому по единице удобно командой КИП4.

Для генерации случайных чисел будем использовать формулу 6. Чтобы она «заработала», надо ввести в калькулятор какое-то начальное случайное число между нулем и единицей: нуль и несколько цифр после запятой. Чтобы образовать такое число, вы можете использовать любые цифры — номер вашего дома и квартиры, номер вашего телефона... А можно посмотреть на часы и написать после запятой часы и минуты: 0.ЧЧММ.

С ввода этого числа в калькулятор пусть и начнется игра. Блок-схема алгоритма и программа приведены на рис. 21, 22. Алгоритм состоит в следующем:

1. Отослать в Р7 введенное случайное число ξ_0 (адрес 00).

2. Вычислить по формуле и отослать в Р7 новое случайное число из диапазона между нулем и единицей (адреса 01—08).

3. На основе вычисленного числа из диапазона]0,1[сформировать целое число из диапазона [10, 99] (адреса 09—27).

4. Образовать числа 10, 99, 100 и заслать их в регистры Р1, Р2, Р3 соответственно; заслать нуль в регистр-счетчик Р4 (адреса 28—38).

5. Прибавить единицу к содержимому Р4 (адрес 39).

6. Образовать и, остановив счет, вывести на индикатор четырехзначное число *АВСД*, где двузначное число *АВ* содержится в Р1 и указывает левую границу промежутка поиска, двузначное число *СД* содержится в Р2 и

указывает правую границу промежутка поиска (адреса 40—45).

7. Ввести во время останова пробное число, запустить калькулятор.

8. Проверить разность между пробным и задуманным числом на равенство нулю (адреса 46—50):

если она равна нулю, перейти к п. 9,

если не равна нулю, перейти к п. 11.

9. Образовать число 55555555 и вывести на индикатор, остановив счет (адреса 51—57).

10. Для продолжения игры вернуться к п. 2 (адрес 58). Заметим, что п. 2 начинается с адреса 01. Именно на этот адрес передает управление команда В/0 в режиме автоматических вычислений по программе. Таким образом, п. 10 можно реализовать единственной командой В/0.

11. Проверить, отрицательна ли разность между пробным и задуманным числом (адреса 59—60):

если отрицательна, перейти к п. 12,

если не отрицательна, перейти к п. 13.

12. Отослать пробное число в Р1 и перейти к п. 5 (адреса 61—64).

13. Отослать пробное число в Р2 и перейти к п. 5 (адреса 65—68).

Попробуйте самостоятельно составить программу, реализующую такой алгоритм, и сравните ее с приведенной здесь (рис. 22).

Порядок игры: перевести переключатель Р-Г в положение Р, ввести программу, вернуться в режим вычислений, набрать произвольное положительное число, не превышающее единицы (например, 0.1234), нажать клавиши В/0 С/П. На индикаторе 1099. Далее набирать пробные числа и нажимать С/П. Каждый раз на индикаторе будет показываться четырехзначное число *ABCD*, где двузначное число *AB* есть левая граница промежутка, содержащего «загаданное» калькулятором число, *CD* — правая. Если пробное число совпадает с «загаданным», на индикаторе 55555555. Для новой игры нажать С/П.

Потренировавшись в отгадывании чисел наедине с микрокалькулятором, вы можете посоревноваться в этом искусстве с приятелями. Быть может, кто-то из вас в конце концов придет к такой уверенной стратегии, принцип которой — деление пополам промежутка поиска.

Интересно, что именно такой прием используется в вычислительных программах, по которым отыскивается

корень непрерывной функции, принимающей на концах некоторого отрезка значения разных знаков. Отрезок делится пополам и из двух его половинок берется для дальнейшего поиска та, на протяжении которой функция меняет знак. Так продолжается до тех пор, пока величина образовавшегося отрезка не станет меньше заданной погрешности вычисления корня. Тогда за его значение принимается любая из точек отрезка.

Но мы опять отклоняемся в сторону от игровой тематики — в сторону исследовательскую, прикладную. Впрочем, это не такое уж отклонение: игры с калькулятором помогают освоить такие приемы работы с ним, которые впоследствии окажутся полезными при решении серьезных научных и практических задач с помощью ЭВМ.

Глава 4

РАЗБИРАЕМ ИГРОВЫЕ ПРОГРАММЫ

Находясь на третьей ступеньке восходящего пути, образуемого главами книги, читатель сможет разобраться в программах некоторых наиболее популярных игр с программируемым микрокалькулятором. В этой главе представлены две из них: «Посадка на Луну» и «Быки и коровы». Для каждой будет подробно изложен алгоритм действий, выполняемых микрокалькулятором в процессе игры, обстоятельно разобрана программа, реализующая этот алгоритм.

ПОСАДКА НА ЛУНУ

Представьте себе, что вы — пилот космической ракеты. Находящаяся на ее борту экспедиция направляется в сторону неизвестной планеты, на которую предстоит совершить посадку.

Двигаясь в безвоздушном пространстве космоса, ракета постепенно приближается к цели. По мере того как расстояние до планеты сокращается, возрастает притяжение, испытываемое с ее стороны ракетой. Ракета падает на планету все быстрее и быстрее.

Чтобы замедлить падение ракеты и мягко посадить ее на поверхность, включаем тормозной двигатель. Задавая различную его тягу, мы тем самым будем управлять торможением нашего космического корабля. Затормозить

мы должны так, чтобы к моменту приземления скорость спуска была не очень большой, такой, которую смогут погасить амортизаторы. Например, не больше 5 м/с, что в земных условиях соответствует падению с высоты, чуть большей 1 м.

Весь этот сложный процесс спуска и торможения ракеты можно промоделировать на микрокалькуляторе. Для этого прежде всего следует сообщить калькулятору все необходимые характеристики нашей ракеты, а также планеты, на которую мы собираемся сесть. Во время игры мы будем управлять работой тормозного двигателя, задавая его тягу на какой-либо интервал времени. А микрокалькулятор по специальной программе (рис. 24) определит, как за этот интервал времени изменится высота ракеты над поверхностью планеты и скорость спуска. Таким образом, процесс спуска моделируется дискретно по тактам (или шагам) вычислений.

Рассмотрим подробнее, какие параметры потребуются микрокалькулятору для расчетов во время игры. Это прежде всего исходные данные: масса нашей ракеты M, кг и запас топлива m, кг; высота H, м, на которой ракета находится над поверхностью планеты в начальный момент времени, и начальная скорость спуска V, м/с. Для того чтобы можно было определить, как меняется сила притяжения к планете по мере приближения к ней, нужно знать ускорение свободного падения у поверхности планеты g_0, м/с^2 и радиус планеты R, м. Тяга тормозного двигателя определяется скоростью вылета раскаленных газов из сопла U, м/с и расходом топлива μ, кг/с. Для упрощения расчетов будем считать две эти величины независимыми. Чтобы экипаж ракеты не подвергался чрезмерным перегрузкам, ускорение, создаваемое двигателем, не должно превышать предельного значения G, м/с^2.

Во время полета нам необходимо следить за показаниями приборов, на которых отображаются такие данные:

высота над поверхностью планеты в текущий момент времени H, м;

скорость спуска V, м/с;

количество оставшегося на борту топлива m, кг;

ускорение a, м/с^2, с которым движется наша ракета;

ускорение свободного падения g, м/с^2 на высоте H;

полное время с начала полета T, с.

Как уже говорилось, работой тормозного двигателя мы будем управлять, задавая расход топлива μ на интервал

времени t. Скорость вылета раскаленных газов U можно принять постоянной.

После того как в микрокалькулятор введены все исходные данные, заданы тяга двигателя и расчетный интервал времени, он вычислит нам новые параметры полета ракеты, достигнутые к концу расчетного интервала. Теперь мы должны решить, как продолжать полет дальше. Скажем, надо ли увеличить тягу двигателя или, наоборот, уменьшить ее. Возможно, нам потребуется не торможение, а дополнительный разгон ракеты в сторону планеты. Это так называемый реверс тяги. Он осуществляется с помощью дополнительного двигателя, расположенного с другой стороны ракеты. На микрокалькуляторе это моделируется так, как если бы направление вылета раскаленных газов сменилось на противоположное. Для этого вводим отрицательное значение U.

Может быть, мы захотим продолжать полет в прежнем режиме, но расчет будет производиться для иного промежутка времени. Вводим в микрокалькулятор новое значение t, оставляя прежними все другие параметры, и затем, после того как будет рассчитан новый этап полета, снова анализируем ситуацию. Так, следя за показаниями навигационных приборов и управляя двигателем, мы подбираем оптимальный режим торможения.

Теперь займемся распределением адресуемых регистров памяти:

A — скорость ракеты V, м/с;
B — высота над поверхностью планеты H, м;
C — запас топлива m, кг;
D — полное ускорение ракеты a, м/с^2;
0 — истекшее с начала полета время T, с;
1 — величина расчетного интервала времени t, с;
2 — расход топлива μ, кг/с;
3 — скорость вылета газов из сопла двигателя U, м/с;
4 — ускорение свободного падения g, м/с^2 на высоте H;
5 — ускорение свободного падения у поверхности планеты g_0, м/с^2;
6 — масса ракеты без топлива M, кг;
7 — радиус планеты R, м.

Регистр 8 — служебный, при расчете очередного этапа полета здесь временно хранится величина запаса топлива, какой она будет к концу этапа. Между промежутками автоматической работы микрокалькулятора, когда на нем можно работать в ручном режиме, этот регистр может быть использован для временного хранения любого дру-

гого числа. Следует учесть, что оно сотрется при расчете нового этапа полета. Регистр 9 в программе не затрагивается. В нем можно длительно хранить какую-либо дополнительную информацию по усмотрению работающего с программой.

Предельная перегрузка G хранится не в регистре, а вводится программно в виде двузначного числа. Оно занимает 13-й и 14-й адреса программы.

Посмотрим теперь, как пользоваться программой. Прежде всего ее нужно ввести в память микрокалькулятора. Для этого устанавливаем счетчик адресов на ноль клавишей В/0. Затем переводим калькулятор в режим программирования клавишей F ПРГ. Вводим текст программы, предварительно определив, какие числа следует поставить по адресам 13 и 14, где должна быть записана предельная перегрузка. Пусть, например, она в пять раз больше, чем ускорение свободного падения у поверхности Земли: $G=5\cdot 9{,}81\approx 49$ м/с^2. Это число и записываем в программную память по отведенным для этого адресам: 13.4 14.9.

По окончании ввода программы переводим микрокалькулятор в режим автоматической работы (F АВТ) и вновь устанавливаем на ноль счетчик команд (В/0).

Вводим исходные данные. Их у нас много, поэтому для ввода остается единственная возможность: набирать очередное число на цифровых клавишах, нажимать затем клавишу П и клавишу, означающую адресуемый регистр. Введем таким образом начальную высоту ракеты над поверхностью планеты в регистр В, начальную скорость спуска — в регистр А. Далее заносим по своим регистрам M, m, U, g_0, R, t. Последним набираем на клавиатуре текущее значение расхода топлива μ. Если мы забудем ввести эту величину, то находящееся в регистре Х число калькулятор примет за нее. Об этом надо помнить перед каждым вводом управляющих параметров на очередной расчетный этап — недосмотр может сорвать продолжение игры.

Нажимаем клавишу С/П, чтобы запустить калькулятор на счет. Примерно через 20 с машина останавливается, и мы можем узнать, как изменились параметры полета нашей ракеты. На индикаторе и в регистре В находится новая высота над поверхностью планеты, в регистре А — новая скорость ракеты. Если необходимо проконтролировать ее или другие параметры, их можно вызвать на индикатор из своих регистров с помощью клавиши ИП.

Для того чтобы приблизить игру к реальным условиям полета, микрокалькулятор выполняет ряд вспомогательных операций, которые имитируют работу контролирующих устройств. Эти устройства следят за действиями пилота и могут запретить выполнение неправильной или опасной команды управления космическим аппаратом. В случае, если сработало одно из таких устройств, происходит аварийный останов и на индикаторе загорается ЕГГОГ.

В том случае, если по ходу нашей игры на индикаторе появилось сообщение об аварийном останове, нужно узнать, по какой причине это произошло. В этом нам поможет число, находящееся к моменту останова в операционном регистре X. Для того чтобы вызвать его на индикатор, выполняем на микрокалькуляторе действие, которое сохраняет неизменным содержимое этого регистра. Например, ↑.А лучше К НОП; при этом неизменным останется содержимое не только регистра X, но и всех других стековых регистров. В нашей игре при возникновении аварийной ситуации мы будем пользоваться именно таким приемом.

Итак, нажимаем клавиши К НОП. На индикаторе появится одно из трех чисел — 1, 2 или 3.

Если на индикаторе загорелась 1, то это означает, что у нас кончается топливо. Его не хватит на работу двигателя в заданном режиме в течение всего расчетного интервала времени. Мы можем дополнительно определить, сколько топлива не хватает. Соответствующее число находится в регистре Y. Нажимаем клавишу ⇄. На индикаторе высвечивается необходимая информация. Для того чтобы продолжить полет, надо уменьшить либо расход топлива, либо интервал времени.

Если в регистре X оказалась 2, то это означает, что расчетное ускорение превысило максимально допустимое. Сработало устройство, защищающее пилота от перегрузки. Если в этом случае с помощью клавиши ⇄ вызвать на индикатор содержимое регистра Y, то высветится величина расчетного ускорения. Иногда бывает, что на протяжении нескольких тактов полет проходит нормально с постоянной тягой, но вдруг в какой-то момент времени ускорение превысило предельное. Это произошло потому, что в результате сгорания топлива уменьшилась полная масса ракеты и прежняя тяга двигателя привела к большему ускорению. Чтобы можно было продолжать полет, приходится уменьшить тягу.

И наконец, 3 в регистре X при аварийном останове означает, что мы недостаточно точно контролируем процесс спуска и задаем слишком большой интервал времени. При этом оказывается, что путь, который должна пройти ракета, превышает расстояние до планеты. Если и в этом случае вызвать на индикатор содержимое регистра Y, то высветится кажущаяся отрицательная высота над поверхностью планеты. Для продолжения полета следует задать меньший интервал времени.

Среди аварийных остановов последнего типа следует особо отметить случай, когда скорость ракеты стала меньше предельной скорости приземления, а расчетный интервал настолько мал, что эта предельная скорость заведомо не будет превзойдена на всем его протяжении. Ракета, стало быть, совершит мягкую посадку на поверхность планеты. Именно в этом случае игру можно считать успешно окончившейся.

Во всех случаях аварийных остановов микрокалькулятор не запоминает неправильно введенные исходные данные, не подходящие для расчета очередного этапа. Заносим новые данные, нажимаем клавишу С/П — и микрокалькулятор заново просчитает тот же этап.

Рассмотрим подробнее, как это происходит. Исходя из данных на текущий момент, введенных величин расчетного интервала времени и расхода топлива, микрокалькулятор, как уже говорилось, определяет, какими окажутся параметры полета к концу этого интервала. Величины, относящиеся к его началу, пометим индексом i, а новые — индексом $i+1$.

Приведем краткий вывод формул, по которым составлена программа вычислений.

Раскаленные газы, вылетающие из сопла ракетного двигателя, развивают силу, численно равную произведению $U\mu_i$ и направленную вертикально вверх. Под действием этой силы ракета приобретает ускорение $U\mu_i/M_i$. Перегрузка, которой подвергается пилот, определяется этой величиной, поэтому необходимо проверить, не превосходит ли она предельного значения G.

Так как на ракету действует еще и сила тяжести, то, следовательно, полное ускорение a_i, с которым будет двигаться наша ракета, составит $U\mu_i/M_i - g_i$. Здесь g_i — ускорение свободного падения на высоте H_i. Определяется оно по формуле $g_i = g_0/(1+H_i/R)^2$.

Введем теперь некоторые допущения. Будем считать, что на протяжении одного интервала времени не изме-

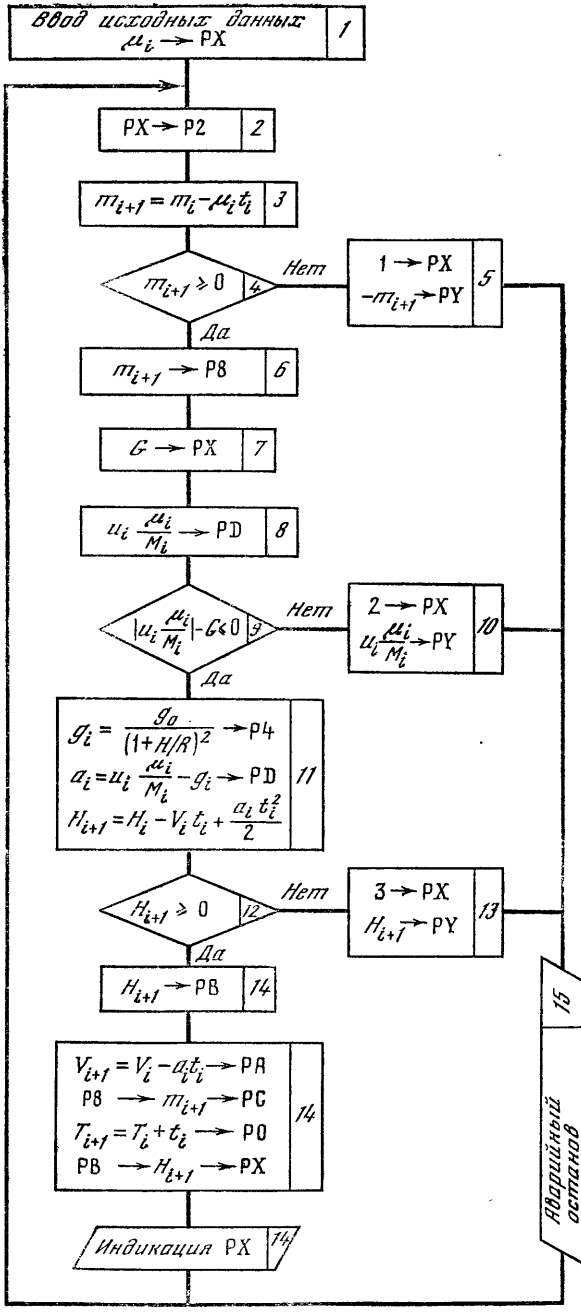

Ввод исходных данных
$\mu_i \rightarrow$ РХ
1
РХ → Р2
2
$m_{i+1} = m_i - \mu_i t_i$
3
$m_{i+1} \geq 0$
4
Нет
1 → РХ
$-m_{i+1} \rightarrow$ РУ
5
Да
$m_{i+1} \rightarrow$ Р8
6
$G \rightarrow$ РХ
7
$u_i \frac{\mu_i}{M_i} \rightarrow$ РD
8
$\left|u_i \frac{\mu_i}{M_i}\right| - G < 0$
9
Нет
2 → РХ
$u_i \frac{\mu_i}{M_i} \rightarrow$ РУ
10
Да
$g_i = \frac{g_0}{(1+H/R)^2} \rightarrow$ Р4
$a_i = u_i \frac{\mu_i}{M_i} - g_i \rightarrow$ РD
$H_{i+1} = H_i - V_i t_i + \frac{a_i t_i^2}{2}$
11
$H_{i+1} \geq 0$
12
Нет
3 → РХ
$H_{i+1} \rightarrow$ РУ
13
Да
$H_{i+1} \rightarrow$ РВ
14
$V_{i+1} = V_i - a_i t_i \rightarrow$ РА
Р8 → $m_{i+1} \rightarrow$ РС
$T_{i+1} = T_i + t_i \rightarrow$ Р0
РВ → $H_{i+1} \rightarrow$ РХ
14
Индикация РХ
14
Аварийный останов
15

Рис. 23

няется полная масса ракеты, а сила тяжести, действующая на ракету, равна произведению ее массы на ускорение свободного падения g_i, существующее на высоте H_i. Кроме того, мы не будем учитывать сопротивление атмосферы планеты. Тогда за интервал времени t_i скорость ракеты изменится на величину $\Delta V = a_i t_i$.

Здесь следует сделать небольшое примечание о знаках, выбираемых при измерении расстояния, пройденного ракетой, ее скорости и ускорения. Расстояние удобно отсчитывать от поверхности планеты так, чтобы положительное направление отсчета было ориентировано от ее центра, отрицательное — к центру. Формально те же направления отсчета следует принять при измерении скорости и ускорения. Опыт показывает, однако, что в нашей игре положительное направление скорости удобнее принять ориентированным к центру планеты. Иными словами, когда ракета приближается к поверхности планеты, ее скорость больше нуля, когда удаляется — меньше. Что же касается ускорения, то тут все, как при измерении расстояний: положительное ускорение направлено от центра планеты, отрицательное — к центру. Будем учитывать это при выводе дальнейших формул.

Полная скорость ракеты на новый момент времени составит $V_{i+1} = V_i - \Delta V = V_i - a_i t_i$. Путь, который пройдет ракета за интервал времени t_i с такой скоростью, равен $\Delta H = V_i t_i - a_i t_i^2/2$. Следовательно, новая высота над поверхностью планеты будет $H_{i+1} = H_i - V_i t_i + a_i t_i^2/2$.

Количество израсходованного топлива составит $\mu_i t_i$. А останется топлива к концу такта $m_{i+1} = m_i - \mu_i t_i$.

Время с начала полета прирастет на величину t_i и составит $T_{i+1} = T_i + t_i$.

По этим формулам составлена программа вычислений для микрокалькулятора. Лежащий в ее основе алгоритм представлен блок-схемой (рис. 23). Поясним отдельные блоки, указывая адреса реализующих их команд.

1. В ручном режиме в соответствующие адресуемые регистры заносятся все исходные данные. Затем на клавиатуре набирается текущее значение расхода топлива μ_i. После этого нажатием клавиши С/П мы пускаем программу на счет.

2. Из регистра X в регистр 2 переписывается только что введенное значение μ_i (00.П2). Напомним: если мы забудем внести этот параметр в РХ перед тем, как нажать клавишу С/П, то последнее находящееся там число калькулятор примет за μ_i.

3. Вычисляется новый остаток топлива m_{i+1} (01.ИПС 02.ИП2 03.ИП1 04.×05.−).

4. Проверяется знак величины m_{i+1} (06.Fx<0 07.12). Если она отрицательна, то топлива не хватит на расчетный интервал времени. В этом случае управление передается на следующий блок 5, где готовится соответствующая информация для вывода на индикатор при аварийном останове. Если же $m_{i+1}\geqslant 0$, то управление передается на адрес 12, начальный адрес блока 6.

5. К моменту останова, вызванного нехваткой топлива, в регистре X должна находиться единица, а в регистре Y — величина нехватки, разность между имеющимся запасом топлива и его планируемым расходом. Такое содержимое регистров X и Y готовится командами 08./−/ 09.1. Следующие команды 10.БП 11.77 передают управление на блок аварийного останова 15.

6. Положительное значение остатка топлива заносится в служебный регистр 8: 12.П8.

7. Командами программы набирается в регистре X значение предельно допустимой перегрузки. Знаки этого числа, например 4 и 9, составляют содержимое этих команд: 13.4 14.9.

8. Вычисляется ускорение $U\mu_i/M_i$, создаваемое двигателем. Затем оно заносится в регистр D на хранение: 15.ИП3 16.ИПС 17.ИП6 18.+ 19.: 20.ИП2 21.× 22.ПД.

9. Определяется модуль ускорения, создаваемого двигателем. Полученная величина $|U\mu_i/M_i|$ вычитается из предельно допустимой перегрузки G (23.Fx^2 24.F$\sqrt{}$ 25.−). Проверяется, не превосходит ли абсолютная величина ускорения, создаваемого двигателем, предельно допустимую перегрузку (26.Fx<0 27.32). Если превосходит, то управление передается на следующий блок 10, где готовится соответствующая информация для вывода на индикатор при аварийном останове. Если не превосходит, управление передается на адрес 32, начальный адрес блока 11.

10. К моменту останова, вызванного чрезмерной перегрузкой, в регистре X должна находиться двойка, в регистре Y — величина $U\mu_i/M_i$. Подготовка обоих операционных регистров выполняется командами 28.FBx 29.2. Следующие команды 30.БП 31.77 передают управление на блок аварийного останова 15.

11. Вычисляется ускорение свободного падения g_i на высоте H_i и заносится в регистр 4 (32.ИПД 33.ИП5 34.ИПВ 35.ИП7 36.: 37.1 38.+ 39Fx^2 40.: 41.П4). Затем

с его помощью определяется полное ускорение ракеты a_i, которое заносится в регистр D (42.— 43.ПД). После этого определяется новая высота над поверхностью планеты H_{i+1} (44.ИП1 45.Fx^2 46.× 47.2 48.: 49.ИПВ 50.+ 51.ИП1 52.ИПА 53.× 54.—).

12. Проверяется знак величины H_{i+1}. Если она отрицательна, управление передается на следующий блок 13, где готовится соответствующая информация для вывода на индикатор при аварийном останове. Если же $H_{i+1} \geqslant 0$, управление передается на адрес 60, начальный адрес блока 14. (55.Fx<0 56.60).

13. К моменту аварийного останова, вызванного тем, что за расчетный интервал времени ракета должна опуститься ниже поверхности планеты, в регистр X засылается тройка (57.3), в регистр Y смещается кажущаяся отрицательная высота, полученная в регистре X при выполнении предыдущего блока. Затем управление передается на блок аварийного останова (58.БП 59.77).

14. Положительное значение высоты H_{i+1} заносится в регистр B (60.ПВ). Вычисляется новая скорость спуска V_{i+1} и заносится в регистр A (61.ИПА 62.ИП1 63.ИПД 64.× 65.— 66.ПА). Из служебного регистра 8 в регистр C переписывается величина нового остатка топлива m_{i+1}, и, наконец, определяется и заносится в регистр 0 полное время с начала полета T_{i+1} (67.ИП8 68.ПС 69.ИП0 70.ИП1 71.+ 72.П0). В регистр X и на индикатор вызывается новая высота над поверхностью планеты, и калькулятор останавливается (73.ИПВ 74.С/П). С клавиатуры вводятся новые управляющие данные на предстоящий этап полета. После нажатия клавиши С/П управление передается на начальный адрес программы, с которого начинается новый цикл вычислений (75.БП 76.00).

15. Блок аварийного останова. Команды 77.ВП 78., вызывают останов, и на индикаторе появляется сообщение ЕГГОГ. Содержимое регистров X и Y при этом не искажается, и находящиеся в них числа выводятся на индикатор соответственно нажатием клавиш К НОП, а затем ⇄.

Рассмотрим конкретный пример игры «Посадка на Луну». На нем мы покажем, как пользоваться программой (рис. 24) и подбирать оптимальный режим торможения. Поиск такого режима с непривычки будет не так уж прост и последователен.

Название игры будем понимать буквально: мы должны мягко посадить свою ракету именно на Луну. Вот ее ха-

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	П2	42	27	32	32	54	–	II
0I	ИПС	6С	28	F Bx	0	55	F x<0	5С
02	ИП2	62	29	2	02	56	60	60
03	ИПI	6I	30	БП	5I	57	3	03
04	×	I2	3I	77	77	58	БП	5I
05	–	II	32	ИПД	6Г	59	77	77
06	F x<0	5С	33	ИП5	65	60	ПВ	4L
07	I2	I2	34	ИПВ	6L	6I	ИПА	6–
08	/-/	0L	35	ИП7	67	62	ИПI	6I
09	I	0I	36	:	I3	63	ИПД	6Г
I0	БП	5I	37	I	0I	64	×	I2
II	77	77	38	+	I0	65	–	II
I2	П8	48	39	F x^2	22	66	ПА	4–
I3	9	09	40	:	I3	67	ИП8	68
I4	9	09	4I	П4	44	68	ПС	4С
I5	ИП3	63	42	–	II	69	ИП0	60
I6	ИПС	6С	43	ПД	4Г	70	ИПI	6I
I7	ИП6	66	44	ИПI	6I	7I	+	I0
I8	+	I0	45	F x^2	22	72	П0	40
I9	:	I3	46	×	I2	73	ИПВ	6L
20	ИП2	62	47	2	02	74	С/П	50
2I	×	I2	48	:	I3	75	БП	5I
22	ПД	4Г	49	ИПВ	6L	76	00	00
23	F x^2	22	50	+	I0	77	БП	0С
24	F $\sqrt{}$	2I	5I	ИПI	6I	78	,	0–
25	–	II	52	ИПА	6–	79	БП	5I
26	F x<0	5С	53	×	I2	80	00	00

Рис. 24.

рактеристики: радиус — 1740 км = $1{,}74 \cdot 10^6$ м, ускорение свободного падения у поверхности — 1,62 м/с^2. Все данные выражены в тех единицах измерения, о которых мы договаривались вначале, их можно так и заносить в предназначенные для них адресуемые регистры: 1,74 ВП 6 П7 1,62 П5.

Предположим, что ракета начинает тормозиться на высоте $H=15$ км над поверхностью Луны и что в этот момент скорость спуска $V=1$ км/с. Вводим эти данные в

микрокалькулятор: 15000 ПВ, 1000 ПА. Масса ракеты без топлива M=500 кг, П6, запас топлива на борту m= =250 кг, ПС. Тормозной двигатель в течение всего процесса спуска будет работать при максимальной скорости выброса раскаленных газов U=3000 м/с, П3.

Предельная перегрузка ограничена величиной, в 5 раз большей ускорения свободного падения у поверхности Земли: 5×9.81=49 м/с². Когда мы станем вводить программу в калькулятор, то по адресам 13 и 14 занесем соответственно 4 и 9.

Все исходные данные заданы, программа введена в микрокалькулятор. Клавишей В/0 устанавливаем на нуль счетчик команд. Можно начинать игру.

Задаем интервал времени 10 с: 10 П1. Устанавливаем подачу топлива μ=15 кг/с. Набираем на клавиатуре число 15. Пускаем программу на счет клавишей С/П... и микрокалькулятор вскоре выдает сообщение об ошибке; на индикаторе загорается ЕГГОГ. В чем дело? Горючее еще не могло кончиться, и до Луны долететь мы еще не могли. Скорее всего остановка произошла из-за того, что мы задали слишком большую тягу. Проверяем, что же на самом деле. Нажимаем клавиши К НОП. На индикаторе загорается цифра 2. Действительно, наши опасения оказались верными, тяга велика. Нажимаем клавишу ⇌. На индикаторе появляется число 60, расчетная величина ускорения. Это больше допустимого предела, равного 49. Надо уменьшить тягу. Введем теперь μ=10 кг/с: 10 С/П.

На этот раз все в порядке: на индикаторе — новая высота над Луной. За 10 с торможения с этой тягой высота нашей ракеты снизилась до 6,9 км (мы округлили результат). Скорость спуска ИП А уменьшилась до 616 м/с. У нас осталось, ИП С, 150 кг горючего. Ускорение, с которым мы тормозились, ИП Д, составляло 38 м/с². Ускорение свободного падения на пройденном нами отрезке пути, ИП4, было почти тем же, что у лунной поверхности: 1,59 м/с².

Пока все вроде бы идет хорошо. На следующие 10 с полета сохраним тот же режим работы двигателя, введем прежнюю величину расхода топлива: 10 С/П. Новая высота над поверхностью Луны H — около 3 км, скорость спуска — 170 м/с. Ускорение, с которым тормозилась наша ракета, теперь оказалось 44,5 м/с². Оно возросло столь резко оттого, что за предыдущие 10 с полета полная масса ракеты M уменьшилась на 100 кг. Горючего у нас осталось всего 50 кг.

Пожалуй, с торможением мы перестарались. До Луны еще порядочно, а скорость уже мала. Чтобы она не упала совсем до нуля и не стала отрицательной (тогда мы будем взлетать), надо либо очень сильно уменьшить тягу, либо на какое-то время совсем выключить двигатель. Так мы и поступим. Набираем 0 и С/П. В течение 10 с мы свободно падаем. Теперь высота стала 1,2 км, скорость спуска возросла до 187 м/с.

Снова включаем двигатель. Только теперь введем меньший интервал времени, поскольку контролировать полет через 10 с — слишком грубо. Зададим t=5 с: 5 П1. Тягу двигателя тоже лучше бы уменьшить. Пусть за эти 5 с расход топлива μ составит 5 кг/с: 5 С/П. Новая высота H=591 м, новая скорость V=58 м/с. Ускорение a=26 м/с^2, горючего m осталось 25 кг.

Скорость опять снизилась очень сильно. Снова уменьшаем тягу. Пусть на следующие 5 с расход топлива μ составляет 3 кг/с.

Рассчитав очередной такт, микрокалькулятор выдает нам H=493 м, V=−19 м/с. Скорость стала отрицательной! Мы явно не рассчитали с тягой. Теперь наш спускаемый аппарат уже нельзя назвать спускаемым — мы взлетаем. Положение усугубляется еще и тем, что горючего осталось всего 10 кг.

В этой ситуации ничего не остается делать, как выключать двигатель и ждать. Ждать придется долго — ускорение силы тяжести на Луне в 6 раз меньше земного, поэтому снова начнем спускаться мы еще не скоро. Вводим прежний интервал времени 10 с, двигатель выключаем: 0 С/П.

За эти 10 с мы поднялись до 605 м, а скорость подъема снизилась до 3 м/с. Ждем еще 10 с. H=555 м, V=+13 м/с. Теперь мы уже спускаемся, но пока еще очень медленно. Еще 10 с будем падать с выключенным двигателем. H=343 м, V=29 м/с. Скорость нарастает. Продолжим еще немного свободное падение. Вводим t=5 с: 5 П1 0 С/П. Новая высота H=177 м, скорость V=37 м/с.

Можно снова включать двигатель. Только теперь будем осторожнее. Необдуманные действия могут плохо кончиться. Попробуем точно рассчитать, как дальше управлять двигателем.

В ручном режиме работы микрокалькулятора прикинем, каким получится ускорение ракеты при μ, равном 1 кг/с. Получаем: $a=U\mu/M-g$=4,3 м/с^2. За 5 с торможения с этой тягой скорость спускаемого аппарата снизится на

величину $at=4{,}3\times5=21$ м/с. Это вполне подходит, так и вводим: 1 С/П. Высота стала $H=43$ м, скорость $V=16$ м/с. Топлива осталось 5 кг.

Пожалуй, тягу можно еще уменьшить. Задаем $\mu=0{,}8$ кг/с. Интервал времени тоже уменьшим, $t=3$ с. Новый такт: $H=9{,}2$ м, $V=6{,}7$ м/с. Топлива осталось 2,6 кг. Ускорение, с которым мы тормозили, составляло около 3 м/с^2.

Пусть ракета тормозится в этом режиме в течение еще 2 с. Скорость при этом уменьшится почти до нуля. В самом деле: $H=2{,}2$ м, $V=0{,}35$ м/с. Снова возникла угроза зависнуть над Луной. Рассчитаем теперь тягу так, чтобы она скомпенсировала силу тяжести, пусть даже и не совсем полностью. Иными словами, тяга может быть настолько малой, что сила тяжести слегка превышает ее. Ракета при этом будет чуть разгоняться, снижаясь. Но это не страшно: ведь сейчас ее скорость очень мала. Пробуем $\mu=0{,}3$ кг/с. Ускорение a оказывается равным 0,18 м/с^2. Положительный знак ускорения говорит о том, что ракета не разгоняется, а тормозится. Это нам не подходит. Тягу надо уменьшить. Пробуем $\mu=0{,}2$. Теперь $a=-0{,}42$. Оно имеет нужное направление, но слишком уж большую величину — мы разгоняемся нежелательно быстро. Чуть увеличим тягу: $\mu=0{,}25$. При этом получается $a=-0{,}12$. Подходит. На этом и остановимся. Интервал времени уже задан (2 с); вводим расход топлива: 0,25 С/П. Теперь $H=1{,}22$ м, $V=0{,}59$ м/с. Так продолжаем тормозить и дальше: 0,25 С/П. И тут микрокалькулятор вновь выдает сообщение об ошибке. Нажимаем К НОП — на индикаторе появляется цифра 3. Понятно: за 2 с мы успеем сесть на Луну. Смотрим содержимое регистра Y, нажимая клавишу $\rightleftarrows$: $H=-0{,}22$. Вводим $t=1$ с и $\mu=0{,}25$ топлива. На сей раз $H=0{,}56$ м, $V=0{,}72$ м/с. Посадка уже почти завершена, и менять управляющие данные нет никакого смысла. Теперь мы просто понаблюдаем окончание спуска. Вводим $t=0{,}5$ с, расход топлива прежний: 0,25 С/П. Считываем с индикатора $H=0{,}18$ м, $V=0{,}78$ м/с.

Можно считать, что мы идеально посадили наш космический корабль на Луну. На высоте $H=18$ см скорость $V=78$ см/с. Это почти вдвое меньше, чем средняя скорость пешехода. Мы даже не почувствуем никакого толчка при посадке. Горючее израсходовано почти полностью. Хорошо однако, что его все же хватило. Торможение заняло 88,5 с. Правда, сам процесс торможения нельзя назвать идеальным. Мы зависали над Луной, взлетали

вверх. Вообще, было совершено довольно много бестолковых действий. Но на первый раз все шло не так плохо.

Надеемся, что в последующих попытках вы добьетесь лучших результатов.

Удачных вам полетов!

БЫКИ И КОРОВЫ

Таково одно из названий этой увлекательной игры. Встречается довольно много ее вариантов, поэтому оговорим правила, взятые за основу при составлении программы для электронного партнера.

С программируемым микрокалькулятором в «быков и коров» можно играть в одиночку. Можно устроить и соревнования. В этом случае каждый из участников проводит с калькулятором несколько партий. Хотя некоторую роль играет и везение, игрок, систематически делающий на основе получаемой информации логичные умозаключения, обычно имеет лучшие результаты.

Правила игры чрезвычайно просты. Один из партнеров — ведущий. Он задумывает любое четырехзначное число. Требуется лишь, чтобы ни одна из составляющих это число цифр не повторялась. Задача другого партнера (будем называть его просто игроком) — отгадать это число. Игрок называет пробные числа. Игра заканчивается, если названо число, задуманное ведущим. Чем меньше ходов-попыток потребуется для этого игроку, тем лучше он играет.

Ведущий помогает игроку подсказками. Что это за подсказки, рассмотрим на конкретном примере.

Предположим, задумано число 9480. Назовем его кодом. Как и предписывается условиями игры, все цифры в коде различны. Игрок называет свое пробное число 7984. Ведущий сравнивает его с кодом 9480. Так как числа не совпадают, то игра не закончена. Ведущий сообщает игроку количество «быков» и «коров», содержащихся в предложенном им пробном числе. «Быками» называются те его цифры, которые по значению и позиции совпадают с соответствующими цифрами задуманного кода. В нашем примере «бык» один — цифра 8. «Коровы» — это цифры, которые совпадают с цифрами задуманного кода, но находятся в иных позициях. В числе 7984 «коров» две: 9 и 4. Понятно, что отгадать код или назвать четыре «быка» — это одно и то же.

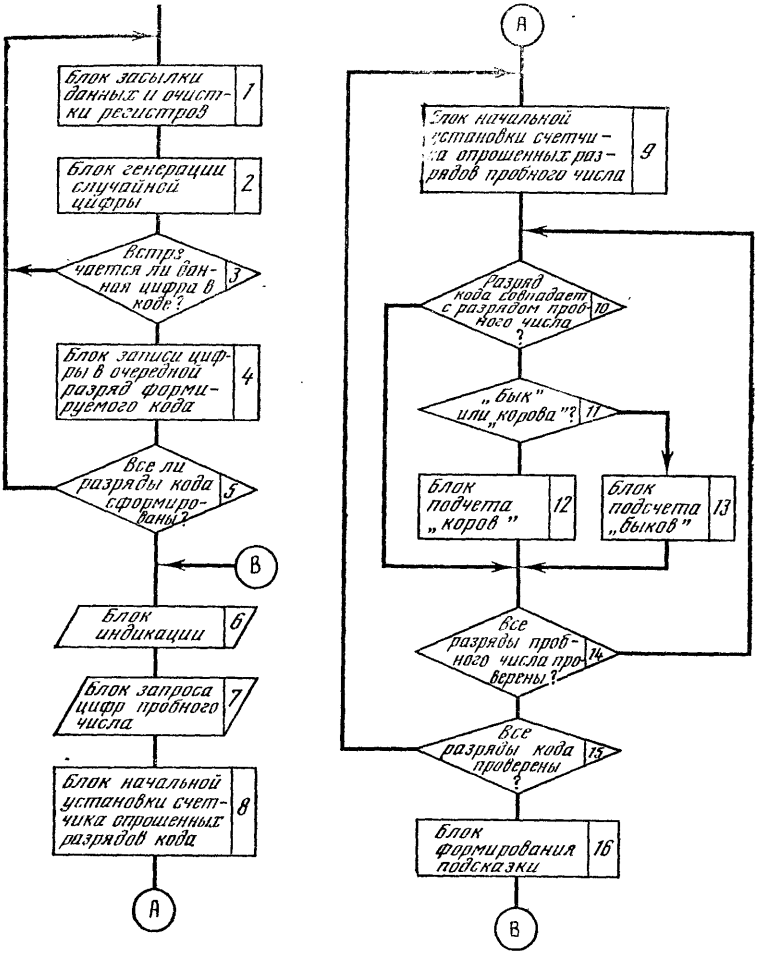


Рис. 25

Если роль ведущего выполняет микрокалькулятор, то его подсказки появляются на индикаторе в виде двузначного числа $БК$, где цифра в старшем разряде $Б$ обозначает количество «быков», цифра в младшем разряде $К$ — количество «коров». Например, если калькулятор «задумал» код 9480, то в ответ на пробное число 7984 на индикатор будет выведено сообщение 12.

Разобраться в программе игры, написанной по изложенному алгоритму, поможет блок-схема (рис. 25).

Что означает каждый из блоков и как они запрограммированы?

Самые первые блоки «задумывают» код, который предлагается игроку для отгадывания. Конечно, микрокалькулятор не думает, а по определенному алгоритму последовательно выдает четыре случайных однозначных числа и, проверив, что все они различны, составляет из них одно четырехзначное число — код. Знаки кода заносятся в адресуемые регистры с 5-го по 8-й включительно: первая цифра — в Р5, вторая — в Р6, третья — в Р7, четвертая — в Р8.

1. Блок засылки начальных данных и очистки регистров для хранения цифр кода. Пропустим пока этот блок — работу его команд лучше пояснить после описания двух следующих блоков.

2. Блок генерации случайной цифры. Делается это в несколько приемов. Сначала получается псевдослучайное число из диапазона]0,1[(вывернутые наружу квадратные скобки здесь означают, что границы диапазона 0 и 1 в него не входят). Для генерации таких чисел в нашей программе используется формула, реализуемая небольшим числом команд, по которой каждое очередное число получается исходя из предыдущего: $\xi_{k+1} = \frac{1}{\pi} \arccos(\sim\cos(10^9 \xi_k))$. Здесь символ $\sim$ означает, что значение косинуса вычисляется микрокалькулятором с погрешностью, причем довольно значительной: она может достигать 100%.

Программа выдает псевдослучайные числа, равномерно распределенные в интервале]0,1[, выполняя команды 09.ИПД 10.ВП 11.9 12.Fcos 13.Farccos 14.Fл 15.: 16.ПД. За ними в программе следуют команды, по которым происходит умножение полученного случайного числа на десять (17.1 18.0 19.×; так мы получаем диапазон]0,10[), добавление единицы (20.1 21.+; получаем диапазон]1,11[) и отделение целой части случайного числа (22.П2 23.КИП2). Обратим внимание: при этом используется команда КИП2. А мы уже знаем: если в команде косвенного вызова вида КИПМ номер регистра *М* равен 0, 1, 2 или 3, то число, находившееся в этом регистре, при выполнении команды КИПМ не только утрачивает дробную часть, но еще и уменьшается на единицу. Это происходит и у нас сейчас. В итоге в Р2 получается случайное однозначное целое число из диапазона от 0 до 9 включительно, иными словами — случайная цифра.

3. Блок проверки условия «Встречается ли данная цифра в коде?». Как уже говорилось, код не должен содержать повторяющихся цифр. Нашей программой цифры кода формируются в обратном порядке — с 4-й, записываемой в Р8, по 1-ю, записываемую в Р5. Понятно, что 4-я цифра может быть при этом любой. Но уже засылая 3-ю цифру в Р7, мы должны проверить, не совпадает ли она с 4-й цифрой, находящейся в Р8; засылая 2-ю в Р6, проверить, не совпадает ли она с 4-й и 3-й, находящимися в Р8 и Р7; засылая 1-ю в Р5, выяснить, не повторяет ли она собою содержимое Р8, Р7 и Р6.

Каждый раз при этом номера регистров перебираются в убывающем порядке. Поэтому адресоваться к ним удобно косвенным образом, с помощью Р0. Занесем в него вначале число 9: 24.9 25.П0. Содержимое регистров с 8-го по 5-й будем вызывать в регистр X для последовательной проверки командой КИП0. Перед каждым ее выполнением содержимое Р0 уменьшается на единицу. Поэтому при первом своем выполнении (9—1=8) она вызовет содержимое Р8, при втором — содержимое Р7 и так далее.

Поскольку проверка содержимого регистров выполняется многократно, блок проверки удобно вынести в подпрограмму: 88.ИП2 89.КИП0 90.— 91.F$x\neq0$ 92.09 93.В/0. Обращение к подпрограмме происходит косвенным образом, командой КППВ. В регистр В для этого предварительно заносится число 88. Первое обращение к подпрограмме командой 26.КППВ вызывает контроль содержимого Р8, второе обращение командой 27.КППВ — контроль Р7, третье 28.КППВ — контроль содержимого Р6.

Рассмотрим работу подпрограммы подробнее. Команда 88.ИП2 вызывает в РХ задуманную цифру, команда 89.КИП0 вызывает содержимое очередного контролируемого регистра. Если результат выполнения операции 90.— равен нулю, значит, цифра повторяется, т. е. текущий регистр проверку не выдержал. В этом случае команды 91.F$x\neq0$ 92.09 передают управление на адрес 09, и блок генерации псевдослучайных чисел выдает новую цифру.

А вот теперь несколько слов о командах 04.П6 05.П7 06.П8, предназначенных для исходной очистки регистров Р6—Р8. Обычно адресуемые регистры очищают, засылая в них нули. Но в нашем случае это не подходит. Поскольку нуль является одной из правомерных цифр кода, то его нельзя использовать для очистки регистров от прежнего содержимого. Заметим, однако, что команды 01.8 02.8 03.ПВ, которые формируют адрес для косвенно-

го обращения к подпрограмме проверки цифр кода, оставляют в регистре X число 88. Это число и записывается в Р6—Р8, вытесняя их прежнее содержимое.

4. Блок записи цифры в очередной разряд формируемого кода. Всего разрядов четыре, поэтому 4 мы и записываем в Р3, счетчик сформированных разрядов кода, командами 07.4 08.П3. Первый разряд, в который требуется записать цифру,— это Р8. Команды 29.ИП3 30.4 31.+ 32.П9 устанавливают в Р9 адрес текущего разряда кода, а именно 8. Остается лишь вызвать подготовленную случайную цифру из Р2 в РХ командой 33.ИП2 и записать ее по установленному адресу командой 34.КП9. При выполнении команды КП9 содержимое Р9 не изменяется.

5. Блок проверки условия «Все ли разряды кода сформированы?». Это один из самых простых блоков. Его составляют команды 35.FL3 36.09. Операция FL3 вычитает из содержимого Р3 единицу, и, если результат не равен нулю, управление передается на адрес, записанный вслед за этой операцией (09). Происходит генерация нового псевдослучайного числа. Если же результат вычитания равен нулю, значит, все разряды кода сформированы и происходит переход на следующий за этими командами адрес, начальный адрес блока 6.

6. Блок индикации подсказки. Сюда мы попадаем, когда подсказка уже сформирована и ее остается лишь продемонстрировать на индикаторе, остановив калькулятор: 38.С/П. Но на эту команду мы попадаем и сразу после того, как сформирован код и игроку предстоит вводить пробное число. Чтобы на индикаторе не оставалась последняя цифра кода, использована команда 37.Сх, очищающая операционный регистр X.

7. Блок запроса цифр пробного числа. После того как на индикаторе появилась подсказка, можно вводить пробное число цифра за цифрой, используя клавишу С/П. Например, если на пробу предлагается число 1234, то следует нажимать 1 С/П 2 С/П 3 С/П 4 С/П. Для хранения цифр пробного числа используются регистры А—Д. В программе данный блок реализуется командами 39.ПА 40.С/П 41.ПВ 42.С/П 43.ПС 44.С/П 45.ПД.

8. Блок начальной установки счетчика опрошенных разрядов кода. Команды 49.4 50.П1 определяют число разрядов кода, равное четырем, а команды 51.8 52.П0 устанавливают начальный адрес опроса.

9. Блок начальной установки счетчика опрошенных разрядов пробного числа, начиная с младшего. Блок этот почти не отличается от блока 8. Команды 53.4 54.П3 определяют число разрядов пробного числа, а команды 55.1 56.4 57.П2 устанавливают адрес младшего разряда пробного числа. Его разряды будут опрашиваться с помощью команды КИП2. Перед первым своим выполнением она вычтет единицу из содержимого Р2, и там получится 13. Мы уже знаем, что калькулятор поймет его как адрес регистра Д.

10. Блок проверки условия «Совпадает ли содержимое текущего разряда кода пробного числа с содержимым текущего разряда кода?». Разряды кода опрашиваются последовательно, и при этом удобно использовать команду косвенного вызова — через регистр 0, в котором (см. блок 8) уже установлен начальный адрес 8.

Выбор Р0 для косвенной адресации может показаться нелогичным: ведь если вызов будет производиться командой КИП0, то она перед каждым своим выполнением станет уменьшать содержимое Р0, и туда, казалось бы, первоначально следует занести не 8, а 9, чтобы при первом выполнении команды КИП0 было вызвано число из Р8, где хранится четвертая цифра кода. Но все дело в том, что в нашей программе используется не команда КИП0, а команда КИП↑, действующая совершенно аналогично, с тем лишь отличием, что при своем выполнении она не изменяет содержимое Р0.

Итак, содержимое текущего разряда кода извлекает команда 58.КИП↑. Текущий разряд пробного числа направляет в регистр X команда 59.КИП2. Команды 60.— 61.Fx=0 62.75 передают управление на блок 11 в том случае, если условие выполняется, и на блок 14 в противном случае.

11. Блок проверки условия «Бык или корова?». Чтобы проверить это условие, достаточно проконтролировать, совпадают ли номера текущих разрядов сравниваемых чисел. Соответствующие команды — 63.ИП3 64.ИП1 65.— 66.Fx=0 67.71. Если номера совпадают, управление передается на блок 13, в противном случае — на блок 12.

12. Блок подсчета «коров». Для накопления числа «коров» используется Р9. Установка его в начальное нулевое состояние производится командами 46.Сх 48.П9. Суммирование организуется командами 71.ИП9

72.1 73.+ 74.П9. Далее управление передается на блок 14.

13. Блок подсчета «быков». В качестве накопителя в данном случае выбран Р4. Предварительно он очищается командами 46.Сх 47.П4. При каждом обращении к нему командой КИП4 его прежнее содержимое увеличивается на единицу. Благодаря этому добавление очередного «быка» осуществляется единственной командой 68.КИП4. Команды 69.БП 70.75 организуют передачу управления на блок 14.

14. Блок проверки условия «Все ли разряды пробного числа проконтролированы?». Блок реализован двумя командами 75.FL3 76.58. Если условие выполняется, то управление передается на блок 15, в противном случае — на блок 10.

15. Блок проверки условия «Все ли разряды кода проверены?». Блок реализован также двумя командами 78.FL1 79.53. Эти команды передают управление на блок 16, если условие выполнено, или на блок 9 в противном случае. Команда 77.КИП0 служит для коррекции адреса текущего разряда кода.

16. Блок формирования подсказки. Как уже отмечалось выше, калькулятор выдает на индикатор подсказку в виде двухразрядного числа БК, где старший разряд Б указывает количество «быков», младший разряд К — число «коров». Соответствующие команды 80.ИП4 81.1 82.0 83.× 84.ИП9 85.+ реализуют формулу 10Б+К. Команды 86.БП и 87.38 передают управление на блок индикации 6.

Нерассмотренной осталась лишь одна команда из блока 1 — команда 00.ПД. Чтобы микрокалькулятор каждый раз задумывал новое число — код, достаточно перед началом игры ввести в РХ какое-то случайное число между нулем и единицей. Откуда его взять? Представьте, что вы собираетесь начать игру. Взгляните на часы. Предположим, они показывают 12 ч 30 мин. В микрокалькулятор следует ввести число 0,1230. Последний нуль, разумеется, можно опустить. Тогда команда 00.ПД направит это число в РД, а программа-генератор псевдослучайных чисел сформирует на его основе цифры кода.

Теперь можно привести полный текст программы (рис. 26), инструкцию и пример игры.

Инструкция:

1. Установите счетчик адресов на нуль клавишей В/0, переведите калькулятор в режим программирования нажатием клавиш F ПРГ, введите программу, верните каль-

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ПД	4Г	32	П9	49	64	ИП1	61
01	8	08	33	ИП2	62	65	-	11
02	8	08	34	КП9	L9	66	Fx=0	5Е
03	ПВ	4L	35	FL3	5-	67	71	71
04	П6	46	36	09	09	68	КИП4	Г4
05	П7	47	37	Cx	0Г	69	БП	51
06	П8	48	38	С/П	50	70	75	75
07	4	04	39	ПА	4-	71	ИП9	69
08	П3	43	40	С/П	50	72	1	01
09	ИПД	6Г	41	ПВ	4L	73	+	10
10	БП	0С	42	С/П	50	74	П9	49
11	9	09	43	ПС	4С	75	FL3	5-
12	Fcos	1Г	44	С/П	50	76	58	58
13	Farccos	1-	45	ПД	4Г	77	КИП0	Г0
14	Fπ	20	46	Cx	0Г	78	FL1	5L
15	:	13	47	П4	44	79	53	53
16	ПД	4Г	48	П9	49	80	ИП4	64
17	1	01	49	4	04	81	1	01
18	0	00	50	П1	41	82	0	00
19	×	12	51	8	08	83	×	12
20	1	01	52	П0	40	84	ИП9	69
21	+	10	53	4	04	85	+	10
22	П2	42	54	П3	43	86	ВП	51
23	КИП2	Г2	55	1	01	87	38	38
24	9	09	56	4	04	88	ИП2	62
25	П0	40	57	П2	42	89	КИП0	Г0
26	КППВ	-L	58	КИП4	ГЕ	90	-	11
27	КППВ	-L	59	КИП2	Г2	91	F x≠0	57
28	КППВ	-L	60	-	11	92	09	09
29	ИП3	63	61	Fx=0	5Е	93	В/0	52
30	4	04	62	75	75			
31	+	10	63	ИП3	63			

Рис. 26

кулятор в режим вычислений нажатием клавиш FABT и вновь клавишей В/0 установите на нуль счетчик адресов.

2. Наберите на клавиатуре число в виде 0,ЧЧММ, где Ч — часы, М — минуты.

3. Запустите программу на счет командой С/П.

4. Нуль, появляющийся на индикаторе примерно через 1—2 мин, означает, что микрокалькулятор «задумал» число и вы можете его отгадывать. Введите пробное число последовательно цифра за цифрой, заканчивая ввод каждой цифры нажатием клавиши С/П.

5. На индикаторе появляется число вида БК, где Б — число «быков», К — «коров». Если на индикаторе появилось число 40 (четыре «быка»), вы отгадали задуманное число, игра успешно закончена. Если хотите сыграть еще раз,— В/О и к п. 2. Если число, задуманное калькулятором, не отгадано, вводите новое пробное число так, как описано в п. 4.

Пример:

В/0, F ПРГ, набираем программу, F АВТ

Показания индикатора	Нажимаем клавиши
	В/0 0,1324 С/П
0	— калькулятор готов к работе; вводим первое пробное число 1234 1 С/П 2 С/П 3 С/П 4 С/П
1	— то, что на индикаторе появилась лишь

одна цифра, означает отсутствие «быков». Цифра показывает число отгаданных «коров». Продолжаем пробовать.

	5 С/П 6 С/П 7 С/П 8 С/П
11	6 С/П 5 С/П 7 С/П 0 С/П
2	5 С/П 7 С/П 6 С/П 9 С/П
21	5 С/П 7 С/П 9 С/П 2 С/П
30	5 С/П 7 С/П 9 С/П 3 С/П
40	число отгадано

Глава 5

СОСТАВЛЯЕМ ИГРОВЫЕ ПРОГРАММЫ

Еще одна глава, еще одна ступенька. По сравнению с предыдущей она существенно выше — здесь собраны особенно сложные игровые программы. Поэтому и разбираются они особенно обстоятельно, показываются в процессе их разработки. Читатель присутствует при этом, вместе с составителями вспоминает по ходу работы раз-

личные приемы программирования на микрокалькуляторе, чтобы в будущем, если захочется, попробовать свои силы в подобном занятии.

БОЙ В ТУМАНЕ

Еще год тому назад Адмирал и не догадывался, что среди самых необходимых ему вещей — компаса, подзорной трубы, карты и трубки — появится калькулятор. А сейчас Адмирал не расстается с ним. Изучив программирование, Адмирал с помощью калькулятора много раз решал важные задачи: выбирал оптимальный курс эскадры, вычислял положение корабля по звездам, рассчитывал наводку артиллерийских орудий. Составив программу-часы, Адмирал смог пользоваться калькулятором как хронометром. И конечно, калькулятор служил Адмиралу как пресс-папье.

Подчиненные Адмирала по-разному относились к калькулятору. Вице-адмирал, верный помощник Адмирала, считал эту машинку чем-то сверхъестественным, написание программы казалось ему чуть ли не колдовством. Кроме того, Вице-адмирал побаивался, что рано или поздно его пошлют в отставку и заменят калькулятором. Контр-адмирал называл калькулятор бесполезной игрушкой. Хорошо зная математику и умея быстро вычислять, Контр-адмирал обходился без калькулятора, хотя и умел им пользоваться.

Однажды эскадра стояла в тумане. Вице-адмирал сообщил тревожные новости:

— В тумане неподалеку от нас находится вражеский корабль. Мы не знаем его координат, он не знает наших. Как только туман рассеется, противник откроет огонь. Чтобы не допустить потерь, предлагаю начать обстрел.

Контр-адмирал: А куда будем стрелять?

Вице-адмирал: По соображениям, неизвестным нам, противник размещает корабли в точках с целочисленными координатами. Будем стрелять по таким точкам вслепую.

Контр-адмирал: Но этих точек многовато! Кроме того, не попадем ли мы в один из своих кораблей?

Адмирал: Почему для определения координат вражеского корабля не используется радиолокация?

Вице-адмирал: Аппаратура вышла из строя. Но у нас есть другая информация — наша радиостанция прослушивает донесения корабля противника. Шифровальщикам

удалось раскрыть код, которым пользуются вражеские связисты. При каждом выстреле противник определяет расстояние от своего корабля до места взрыва снаряда и передает его. Можно ли, сделав несколько выстрелов и услышав несколько подобных донесений, узнать координаты корабля?

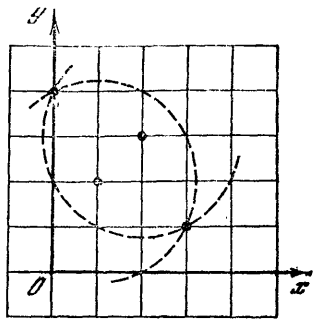


Рис. 27

Адмирал: Да, можно. Сейчас я составлю и введу в калькулятор программу «Корабль противника». Пусть объект имеет координаты X и Y: эти числа мы перед пуском программы введем в адресуемые регистры D и C. Стрелять будем так: координаты точки прицеливания x и y вводятся в операционные регистры X и Y соответственно; программа вычисляет и сообщает нам расстояние между кораблем и взрывом: $S=\sqrt{(X-x)^2+(Y-y)^2}$. Если корабль поражен, $S=0$. Если $S\neq 0$, можно выстрелить в другую точку.

Адмирал написал программу:

00.ИПД 01.− 02.Fx^2 03.⇄ 04.ИПС 05.− 06.Fx^2 07.+ 08.F$\sqrt{}$ 09.С/П

Адмирал: Вице-адмирал, введите в калькулятор условные координаты корабля противника: X ПД Y ПС. Контр-адмирал, попробуйте выяснить их. Введите координаты точки, в которой разорвался ваш снаряд, ну, скажем, $x=1$, $y=2$.

Контр-адмирал набрал 2 ↑ 1 В/О С/П. Вскоре на индикаторе появился результат: $S=2{,}2360678$.

Контр-адмирал: А как произвести следующий выстрел?

Адмирал: Пока калькулятор стоит, введите координаты новой точки прицеливания. Нажатием клавиши В/О передайте управление на адрес 00. Нажатием клавиши С/П запустите калькулятор. Он выполнит команды программы с новыми исходными данными и остановится по команде 09.С/П с новым расстоянием S на индикаторе. Так, исследуя нашу модель, мы найдем способ поразить противника!

Контр-адмирал: Можно обойтись и без модели. Я уже знаю более простой способ поиска. После выстрела в точку (1,2) мне известно, что корабль находится на рас-

стоянии $\sqrt{5}$ от этой точки. Отметим на карте точку (1,2) и проведем окружность с центром в ней и радиусом $\sqrt{5}$. Корабль находится на этой окружности. Сделаем второй выстрел — (2,3). Результат... $S=\sqrt{5}$. Проводим окружность с центром в точке (2,3) и тем же радиусом, что и предыдущую (рис. 27).

Окружности имеют две точки пересечения: (0,4) и (3,1). В одной из них — корабль. Двумя выстрелами мы наверняка поразим его. Таким образом, с помощью простейших геометрических построений мы определим положение корабля и поразим его за четыре выстрела.

Вице-адмирал: В реальной обстановке корабль может не стоять на месте, ожидая наших выстрелов, а двигаться вдоль некоторой прямой; при этом за время между двумя выстрелами он переходит в соседнюю точку с целочисленными координатами $X+\Delta X$, $Y+\Delta Y$ либо по вертикали, либо по горизонтали, либо по диагонали, т. е. движется «ходом шахматного короля».

Адмирал: Мы построим модель и для этого случая. Кроме того, мы сделаем нашу программу удобнее. Автоматизируем ввод исходных числовых данных. Мы ведь знаем, что при нажатии клавиши ПП в режиме ручных вычислений калькулятор выполняет очередную команду программы и останавливается. Напишем в начале программы команды засылки ПД, ПС, ПВ, ПА. Установив счетчик адресов на нуль клавишей В/0, наберем на клавиатуре значение X_0 и нажмем клавишу ПП. Калькулятор выполнит стоящую по адресу 00 команду ПД, по которой введенная величина будет послана в регистр Д, и остановится. Пока он стоит, наберем на клавиатуре величину Y_0 и опять нажмем клавишу ПП. Будет выполнена следующая команда программы, отсылающая введенную величину в регистр С... Так же введем далее величины ΔX и ΔY: они окажутся на своих местах, в регистрах В и А.

Мы организуем в нашей программе выдачу сообщения SOS (505), если $S=0$ и, стало быть корабль поражен; еще сделаем так, чтобы продолжать счет без многократных нажатий клавиши В/0. В этом нам помогут безусловные и условные переходы. Алгоритм несложен:

1. Ввести $X_0\to$РД, $Y_0\to$РС, $\Delta X\to$РВ, $\Delta Y\to$РА.
2. Повторять действия (до тех пор, пока $S\neq 0$):
   2.1. Остановить программу для индикации S и ввода x, y.
   2.2. Вычислить $S=\sqrt{(X_h-x)^2+(Y_h-y)^2}$.

2.3. Вычислить $X_{k+1}=X_k+\Delta X_k$, $Y_{k+1}=Y_k+\Delta Y_k$.
3. Вывести сообщение SOS.
4. Закончить работу.

Затем Адмирал написал программу:

00. ПД	08. Fx^2	16. ИПД	24. F○
01. ПС	09. ⇄	17. +	25. F$x=0$
02. ПВ	10. ИПС	18. ПД	26. 05
03. ПА	11. —	19. ИПА	27. 5
04. Сх	12. Fx^2	20. ИПС	28. 0
05. С/П	13. +	21. +	29. 5
06. ИПД	14. F$\sqrt{}$	22. ПС	30. С/П
07. —	15. ИПВ	23. F○	31. БП
			32. 00

— Адреса 00—04 — это ввод исходных данных,— пояснил Адмирал.— Останов по адресу 05 сделан для того, чтобы узнать расстояние S от точки разрыва снаряда до вражеского корабля — оно находится в регистре X и на индикаторе, когда по ходу счета мы попадаем на этот адрес,— и для ввода новых координат прицеливания. Адреса 06—14: определение расстояния S; 15—22: вычисление нового положения корабля; 23—24: вызов S из регистра Z в регистр X; 25—26: проверка, не попал ли выпущенный снаряд во вражеский корабль, и в случае промаха — переход на адрес 05, где мы выстрелим вновь; 27—30: сюда передается управление в случае попадания, здесь из цифр 5 0 5 формируется сообщение SOS и демонстрируется на индикаторе после останова; 31—32: запустив калькулятор вновь, мы переходим к командам по адресам 00—04, которым вводится информация о движении другого вражеского корабля, если он есть, чтобы открыть стрельбу по нему.

После того как программа была введена в микрокалькулятор, Адмирал ввел исходные данные X_0 ПП Y_0 ПП ΔX ПП ΔY С/П, используя клавишу ПП для поочередного выполнения команд 00—03. Вице-адмирал принялся вводить координаты точек прицеливания x и y, пытаясь определить положение вражеского корабля, но задача оказалась сложной.

Контр-адмирал: Так можно стрелять до бесконечности, но ведь на самом деле боезапас ограничен. Модель этого не учитывает.

Адмирал: Модель, конечно, можно усовершенствовать. В прошлом примере мы организовали программу так, что ее выполнение продолжалось до тех пор, пока имело место

условие $S \neq 0$. Такая конструкция называется циклом с условием. Мы воспользуемся другим видом цикла — циклом с параметром. Это конструкция «для всех значений k от M до N выполнять действия...». Давайте ограничим число выстрелов N и организуем возврат к началу цикла лишь в том случае, если боезапас не исчерпан. Если снаряды израсходованы, а противник цел, мы получим сообщение ЕГГОГ. Здесь нам пригодится операция FL1.

Вице-адмирал: А что это за операция?

Адмирал: Работает она так: из содержимого первой ячейки (параметра цикла) вычитается единица. Если результат не равен нулю, то он записывается в Р1, и происходит переход по адресу, указанному после операции FL1. Если после вычитания результат равен нулю, то он не засылается в Р1, и там остается единица, а программа выполняется далее с команды, следующей после адреса перехода.

Когда надо k раз подряд выполнить несколько команд, цикл организуется в виде такой последовательности команд:

1. Команды, записывающие число k в Р1.
2. Команды, которые необходимо повторять (тело цикла).
3. Операция FL1.
4. Адрес начала тела цикла.
5. Команды, следующие после цикла.

Каждый раз, когда выполнены команды тела цикла, операция FL1 уменьшает параметр цикла на единицу и, если он не равен нулю, обеспечивает переход для повторного выполнения цикла. Ту же роль, что регистр 1, могут сыграть регистры 0, 2, 3, но вместо операции FL1 тогда надо использовать FL0, FL2 и FL3 соответственно.

Впрочем, мы отвлеклись. На чем я остановился? Да, на алгоритме! Так вот, в нашей задаче алгоритм имеет вид:

1. Ввести $X_0 \to$ РД, $Y_0 \to$ РС, $\Delta X \to$ РВ, $\Delta Y \to$ РА, $N \to$ Р1.
2. Повторять действия (для всех k от 1 до N):
   2.1. Остановить программу для индикации S и ввода x, y.
   2.2. Вычислить $S=\sqrt{(X_k-x)^2+(Y_k-y)^2}$.
   2.3. Вычислить $X_{k+1}=X_k+\Delta X_k$, $Y_{k+1}=Y_k+\Delta Y_k$.
   2.4. Если $S=0$, то:
      2.4.1. Выдать сообщение SOS.

2.4.2. Закончить работу.
3. Выдать сообщение ЕГГОГ.
4. Закончить работу.
Программа:

00. ПД	19—22 ИПА ИПС + ПС
01. ПС	23—24 F⊙ F○
02. ПВ	25—26 F$x\neq0$ 33
03. ПА	27—28 FL1 05
04. П1	29. К—
05. С/П	30. С/П
06—14 ИПД — Fx^2⇄ ИПС —Fx^2 + F$\sqrt{}$	31—32 БП 00
15—18 ИПВ ИПД + ПД	33—37 5 0 5 БП 30

Адмирал: Заметьте, что в этой программе многие блоки взяты из предыдущей. Адреса 00—04 — ввод исходных данных, изменилась лишь последняя из этих команд, 04.П1 — ею в регистр П1 записывается число повторений цикла. Команды по адресам 05—26 такие же, как раньше, изменился только адрес перехода после операции F$x\neq0$. Адреса 27—28 — это операция организации цикла FL1 и адрес перехода 05 на команды вывода S и ввода координат точки прицеливания. Если боезапас израсходован, мы переходим на следующий адрес 29, на команду К—. Она не имеет смысла, калькулятор останавливается по ней и выводит на индикатор сообщение ЕГГОГ. Если после этого запустить калькулятор клавишей С/П, он перескочит через адрес (так уж действует команда К—), т. е. перейдет к команде 31.БП 32.00 и согласно ей вернется к началу программы, чтобы принять информацию для последующего обстрела. Мы, конечно, совершим этот переход по шагам, с помощью клавиши ПП. Ну а адреса 33—37, 30 — это формирование сообщения SOS о попадании и останов, чтобы прочесть это сообщение. Если после этого мы вновь запустим калькулятор, то перейдем на команды ввода новой информации.

Введем программу, а затем данные: X_0 ПП Y_0 ПП ΔX ПП ΔY ПП N С/П $y\uparrow x$ С/П $y\uparrow x$ С/П...

Пока Контр-адмирал пытался найти общий метод поиска кораблей, Вице-адмирал думал над более сложным вопросом.

Вице-адмирал: А если, пытаясь нас обмануть, корабль станет лавировать, двигаться «ходом шахматного короля» не по прямой, а по ломаной, причем приращения коор-

динат ΔX_m и ΔY_m станут изменяться с периодом T=3, 4 или 5?

После первого выстрела координаты корабля получат приращения ΔX_1 и ΔY_1, после второго — ΔX_2 и ΔY_2 и так далее; после некоторого приращения их последовательность повторяется. Можно ли, используя алгоритм предыдущей программы, написать новую?

Адмирал: Пожалуй, да. Я покажу это на примере для T=5.

1. Ввод данных.

2. Ввод координат выстрела; вычисление расстояния от взрыва до корабля; проверка, произошло ли попадание и израсходован ли боезапас; координаты корабля получают приращения ΔX_1 и ΔY_1.

3. Обработка 2-го выстрела аналогично 1-му и изменение координат корабля на ΔX_2 и ΔY_2. ...

6. Обработка 5-го выстрела аналогично предыдущему и изменение координат корабля на ΔX_5 и ΔY_5.

7. Переход к п. 2.

8. Сообщение о попадании в корабль.

9. Сообщение об израсходовании боезапаса.

В п.п. 2—6 различаются лишь адреса ΔX_m и ΔY_m, а в остальном они совпадают.

Контр-адмирал: Написать такую программу, конечно, можно, но в память микрокалькулятора она не поместится. При T=5 она займет примерно 150 команд!

Адмирал: Заметьте, что большие части этой программы будут пять раз повторяться. Эти части можно оформить как подпрограмму, а в главной программе организовать обращения к ней. Тогда программа станет гораздо короче:

```
00—12 ПД ПС ПВ ПА П9 П8 П7 П6 П5 П4 П3
      П2 П1
13—14 ПП 69

15—17 ИПВ ПП 47      30—32 ИП6 ПП 51
18—20 ИПА ПП 51      33—35 ИП5 ПП 47
21—23 ИП9 ПП 47      36—38 ИП4 ПП 51
24—26 ИП8 ПП 51      39—41 ИП3 ПП 47
27—29 ИП7 ПП 47      42—44 ИП2 ПП 51
                     45—46 БП 15
```

Адреса 00—12 — ввод исходных данных (о порядке ввода — ниже), адреса 13—14 — переход на подпрограмму, где на индикатор выводится расстояние S и вводятся координаты точки прицеливания; каждая из последующих

троек команд, оканчивающаяся переходом на адрес 47,— это подготовка данных для подпрограммы, вычисляющей координату X_{k+1}, и обращение к этой подпрограмме; каждая из троек, оканчивающаяся переходом на адрес 51,— подготовка данных для подпрограммы, вычисляющей координату Y_{k+1} и выдающей результаты выстрела, и обращение к этой подпрограмме.

А вот и сами подпрограммы:

47.ИПД 48.+ 49.ПД 50.В/0

Как видите, координата X_{k+1} вычисляется теми же операциями, что и прежде.

51. ИПС	59.69	66.5	73. $\rightleftarrows$
52. +	60. К—	67. БП	74. ИПС
53. ПС	61. С/П	68.61	75. —
54. F$\bigcirc$	62.БП	69. С/П	76.Fx^2
55. F$\odot$	63.00	70. ИПД	77. +
56. F$x \neq 0$	64.5	71. —	78.F$\sqrt{}$
57.64	65.0	72. Fx^2	79.В/0
58. FLI			

И здесь все по-старому — можете проверить. Порядок ввода данных: X_0 ПП Y_0 ПП ΔX_1 ПП ΔY_1 ПП ΔX_2 ПП ΔY_2 ПП ΔX_3 ПП ΔY_3 ПП ΔX_4 ПП ΔY_4 ПП ΔX_5 ПП ΔY_5 ПП N С/П $y\uparrow x$ С/П $y\uparrow x$ С/П...

Вице-адмирал: По-моему, здесь можно организовать цикл!

Адмирал: Наиболее разумно это можно сделать, используя косвенную адресацию. Она позволит перебрать ΔX_m и ΔY_m, изменяя индекс m. Кроме того, косвенная адресация укоротит блок ввода данных — я покажу это в свое время, чуть позже. Мой алгоритм:

1. Для L от 13 до 1:
   1.1. Ввести в регистр с номером L число из ряда (X_0; Y_0; ΔX_1; ΔY_1; ΔX_2; ΔY_2; ΔX_3; ΔY_3; ΔX_4; ΔY_4; ΔX_5; ΔY_5).
2. Повторять (пока $N>0$):
   2.1. Остановить программу для индикации S и ввода X, Y.
   2.2. Вычислить $S=\sqrt{(X_k-x)^2+(Y_k-y)^2}$.
   2.3. Если $S=0$, то
      2.3.1. Выдать сообщение SOS.
      2.3.2. Закончить работу.

В противном случае:

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	I	0I	I7	Fx≠0	57	34	КИП0	I0
0I	3	03	I8	45	45	35	ИПС	6C
02	П0	40	I9	ИП0	60	36	+	I0
03	С/П	50	20	3	03	37	ПС	4C
04	КП↑ 0	LE	2I	-	II	38	Г0	25
05	FL0	5Г	22	F x<0	5C	39	FLI	5L
06	03	03	23	28	28	40	07	07
07	С/П	50	24	I	0I	4I	К-	27
08	ИПД	6Г	25	2	02	42	С/П	50
09	-	II	26	П0	40	43	БП	5I
I0	F x^2	22	27	Г0	25	44	00	00
II	⇄	I4	28	Г0	25	45	5	05
I2	ИПС	6C	29	КИП0	I0	46	0	00
I3	-	II	30	ИПД	6Г	47	5	05
I4	F x^2	22	3I	+	I0	48	БП	5I
I5	+	I0	32	ПД	4Г	49	42	42
I6	F√	2I	33	Г0	25			

Рис. 28

2.3.3. Если счетчик индексов не соответствует требуемым ячейкам, то

2.3.3.1. Обновить счетчик индексов.

2.3.4. Вычислить $X_{k+1}=X_k+\Delta X_k$, $Y_{k+1}=Y_k+\Delta Y_k$.

2.3.5. Уменьшить N на 1.

3. Выдать сообщение о том, что боезапас исчерпан.

4. Закончить работу.

Программа — на рис. 28.

— Команды 00.1 01.3 02.П0,— пояснил Адмирал,— засылают число 13 в регистр-счетчик 0, через который будет производиться косвенная засылка начальных данных. Сразу после этого — останов 03.С/П. Набираем на клавиатуре первое из ряда засылаемых чисел. Следующая команда КП↑ действует почти так же, как команда КП0: она пересылает содержимое регистра X в регистр, номер которого выражается содержимым регистра 0.

Вице-адмирал: Но там содержится число 13, а у нас нет регистра с номером больше 9-го!

Адмирал: Если вы продолжите нумерацию на регистры с буквенными обозначениями, то получите, что 10 — это номер регистра A, 11 — регистра B, 12 — регистра C, 13 — регистра D. Именно так расшифровываются числа с 10 по 13 при косвенной адресации.

Контр-адмирал: Но команда КП0 вычитает единицу из содержимого регистра 0 и посылает содержимое регистра X в регистр с номером, равным полученной разности.

Адмирал: Повторяю, у нас в программе не команда КП0, а команда КП↑, отличающаяся от КП0 тем, что не изменяет содержимого регистра 0. Поэтому первое из ряда засылаемых чисел направляется в Р13, т. е. в РД. А вот следующая команда 05. FL0 06.03 уже уменьшит содержимое регистра 0, там получится 12, и, поскольку это число еще не равно единице, управление будет передано на адрес 03, на команду останова. Мы наберем на клавиатуре второе из ряда вводимых чисел, и команда КП↑ направит его в регистр 12, т. е. в регистр C. Так все числа ряда будут расставлены по своим регистрам в убывающем порядке номеров, с 13-го по 1-й (в нем, напомню, у нас хранится общее число выстрелов N). Заметьте: это очень удобный способ для засылки большого массива чисел в адресуемые регистры.

Когда засылка закончится, калькулятор остановится по команде 07.С/П. Мы введем координаты выстрела. Следующими командами, записанными по адресам 08—16, вычисляется расстояние S. Если вы проследите адреса 17—28, то увидите, что после исполнения этих команд в регистре 0 окажется число 12, в регистре X — расстояние S. Если с помощью таблицы, показывающей движение чисел по стеку, вы разберете работу команд по дальнейшим адресам 29—38, то увидите, что команда 29.КИП0 вызывает из регистра B величину ΔX_1, команда 34.КИП0 извлекает из регистра C величину ΔY_1, команды 33.FO и 38.FO сдвигают числа вниз по стеку, чтобы он каждый раз был готов к приему нового вызываемого числа, вся цепочка команд по адресам 29—38 вычисляет новое положение вражеского корабля в соответствии с нашими формулами, а следующие две команды 39.FL1 40.07 отсылают нас N раз подряд к команде останова 07.С/П, где на индикатор выводится расстояние S и вводятся координаты нового выстрела... Но я что-то слишком уж

подробно объясняю. Вы сможете сами разобраться во всей программе до конца, тем более что заключительные команды по адресам 41—49 повторяют окончание прежней программы вплоть до перехода 48.БП 49.42.

Прервав свой рассказ и подождав, пока Вице-адмирал и Контр-адмирал внимательно ознакомятся с его программой, Адмирал ввел в калькулятор ее и затем исходные данные: Сх С/П X_0 С/П Y_0 С/П ΔX_1 С/П ΔY_1 С/П ΔX_2 С/П ΔY_2 С/П ΔX_3 С/П ΔY_3 С/П ΔX_4 С/П ΔY_4 С/П ΔX_5 С/П ΔY_5 С/П N С/П.

Вице-адмирал и Контр-адмирал открыли стрельбу, вводя в регистры стека координаты точек прицеливания: $y\uparrow x$ С/П. После пуска программа Адмирала вычислила расстояние от корабля до взрыва, проверила, попал ли снаряд в корабль, поменяла в соответствии с заданным законом координаты корабля, проверила, не израсходованы ли все снаряды и сообщила адмиралам расстояние от корабля до взрыва. Обстрел продолжался: $y\uparrow x$ С/П (на индикаторе S), $y\uparrow x$ С/П...

Грохот разрывающихся снарядов доносился издалека и сливался с ревом бурного моря. Какой-то резкий звон вдруг ворвался в этот шум; гул разбушевавшихся волн стал постепенно стихать, и на этом слабеющем фоне еще явственнее слышались звуки разрывов...

Отважные Адмиралы так увлеклись математической моделью боя, что не заметили настоящей опасности, которая приближалась к ним. Опасность приблизилась и сказала:

— Почему вы на уроке не слушаете учителя? Разве вы не слышали звонка об окончании перемены?

МОРСКОЙ БОЙ

Как вы уже, наверное, догадались, отважные Адмиралы — еще школьники. Они учатся в 9-м классе. Сохраним для истории их имена: Адмирала зовут Сергеем, Вице-адмирала — Константином, Контр-адмирала — Николаем.

«Бой в тумане», во время которого мы познакомились с ними, скорее походил на учебные стрельбы, чем на настоящий «Морской бой». К тому же наш рассказ пестрил подробностями учебного характера, многие из которых читателю этой книги наверняка помнятся по предыдущим главам. Но недаром говорят: повторенье — мать ученья. Бравым Адмиралам предстояло изучить еще многое по части программирования на микрокалькуля-

торе, прежде чем приступить к составлению программы для «Морского боя». Какой бы привычной и несложной ни казалась эта игра, требуется немало сообразительности, чтобы перевести ее правила и искусство играть в нее на язык калькуляторных команд.

Некоторое время тому назад у Николая появился собственный микрокалькулятор. Часто Сергей с Николаем собираются вместе, чтобы общими усилиями решить сложную задачу на своих карманных ЭВМ. Константин наконец решился осваивать программирование и, узнав, что его приятели собираются писать программу для настоящего «Морского боя», попросил Сергея позвать его, когда они с Николаем примутся за программирование.

О том, как это происходило, пусть Костя расскажет сам.

— Мы договорились встретиться у Сергея (Адмирал оставался Адмиралом) в четверг после уроков. Я вооружился блокнотом, шариковой ручкой, и ровно в 16.00 мы уже сидели за столом в кают-компании (гостиной). Посередине стола красовались два микрокалькулятора: «Электроника БЗ-34» и «Электроника МК-54». Хотя внешне машинки мало походили друг на друга, оказалось, что различаются у них только обозначения некоторых клавиш.

— Ну, с чего начнем? — нетерпеливо поинтересовался Коля.

— Надо вспомнить в точности все правила игры,— предложил Сережа.

— Можно мне? — я решил поучаствовать.— Играем вдвоем. Каждый берет чистый листок в клеточку и на нем рисует два квадрата — игровые поля размером 10×10. Строки и столбцы поля кодируем буквами и цифрами. У меня на листке первое поле — мое. На нем я расставляю свои корабли. Всего их, допустим, двадцать.

Я старался не упустить ни одной детали:

— У противника такой же листок, такие же два поля, и на первом — его корабли. Второе поле на моем листке — это поле противника. Тут я буду рисовать, как стоят его корабли. Я называю букву и цифру — координаты поля, по которому я стреляю. Противник говорит, попал я или промахнулся, и я ставлю на втором поле крестик или точку соответственно. Потом стреляет противник и так же ставит на своем листке, на втором поле, крестик или точку. Если я уничтожу его корабли раньше, чем он мои, я выиграл. Да, чуть не забыл еще одно правило:

при удачном выстреле, т. е. при попадании в корабль противника, можно сделать еще один выстрел. Вот, по-моему, все.

— Все точно,— подтвердил Коля.— Ну что, примемся за алгоритм?

— Нет, прежде чем заниматься алгоритмом, давайте решим, как мы будем изображать игровое поле. Как ты думаешь, сколько регистров понадобится для этого?— обратился Сергей к Коле.

— Сейчас, сейчас... Если, например, каждую клеточку игрового поля представить отдельным регистром, то самое большое квадратное поле, которое мы сможем «запомнить», будет состоять всего из девяти клеток, т. е. 3×3. Не-е-ет, ничего не получится,— разочарованно протянул Коля,— в такой «Морской бой» только малышам играть. Даже поле 4×4 требует 16 регистров, а в микрокалькуляторе их всего 14.

— Неужели нельзя по-другому хранить числа в памяти машины?— огорчился я.

— Конечно же, можно!— успокоил нас Сергей.— Коля, ты же знаешь, что микрокалькулятор работает с восьмиразрядными числами?

— Знаю, разумеется,— подтвердил Коля.

— Так вот, в любом разряде такого числа может встретиться любая цифра от 1 до 9. Таким образом, в одном разряде — одна клетка игрового поля, в восьми разрядах — восемь...

— Тогда калькулятор сможет запомнить поле 8×8, и еще останутся свободные регистры для программы,— Коля повеселел.— Ну что, пора приниматься за алгоритм. С чего же начать?

— Вспомним еще раз правила. Игровым полем чаще всего служит чистый листок в клеточку. На листке чертят два поля размером...— начал Сергей.

— Стоп, стоп,— прервал его Коля,— уже можно рисовать блок-схему. Где карандаш? Вот первый квадратик.

Очистка игрового поля

Что там дальше? По определенным правилам расставляем несколько кораблей на «своем» поле.

Расстановка кораблей

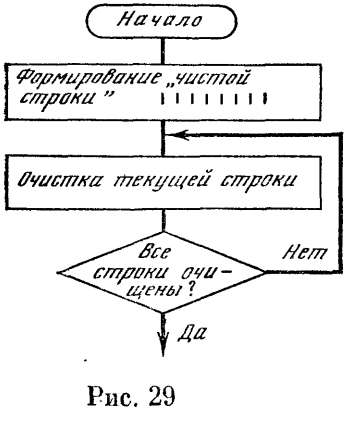


Рис. 29

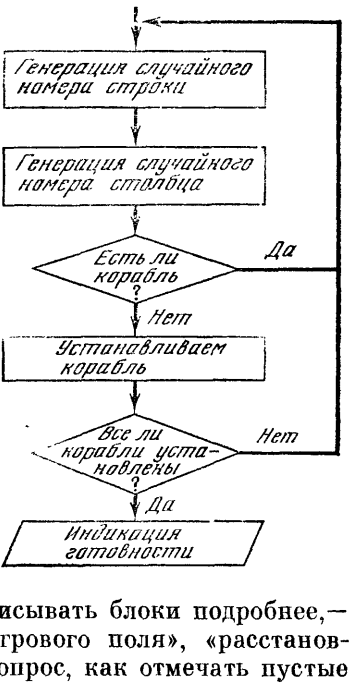


Рис. 30

— Нет, стоит, по-моему, описывать блоки подробнее,— возразил Сергей.— «Очистка игрового поля», «расстановка кораблей»... Надо решить вопрос, как отмечать пустые клеточки, а как — корабли.

— Знаете, ребята, раз уж мы выбрали поразрядную обработку, то самое простое решение — использовать символы 0 и 1, как в двоичном коде: 0 — нет корабля, 1 — корабль,— предложил я.

— Сама идея интересна, но для калькулятора неприменима,— не согласился Сергей.

— Если, например, на индикаторе 10000000,— стал объяснять мне Коля,— все хорошо. Мы можем договориться, что в левой позиции этой строки игрового поля стоит корабль. А представь себе такое расположение корабля.— Коля включил свой микрокалькулятор и быстро набрал число 00010000.— Те нули, что слева от 1, называются незначащими. Для их гашения в микрокалькуляторе предусмотрена специальная схема, помнишь, нам это учитель физики объяснял?

Коля нажал кнопку ↑, и число на индикаторе обрело привычный, но, увы, не подходящий для игры вид 10000.

— Что же делать? — спросил я.

— Придется заменить 0 на 1, а 1, например, на 2. Теперь код вида 11111112 будет показывать, что корабль

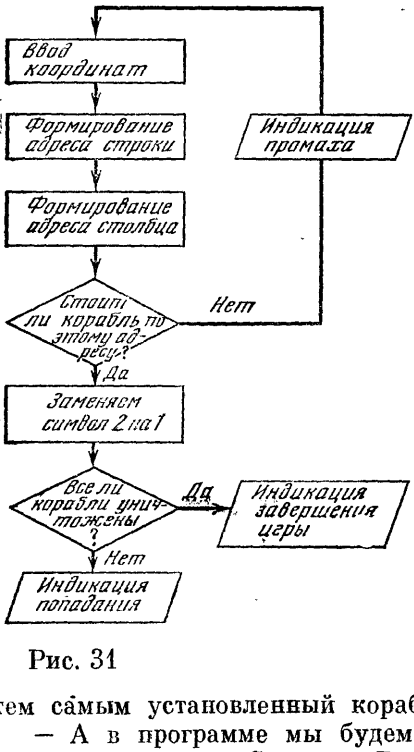


Рис. 31

стоит в самой «правой» клеточке одной из строк игрового поля.

— Теперь я могу нарисовать кусочек подробной блок-схемы,— Коля взял в руки карандаш (рис. 29).

— А как с расстановкой кораблей? — закончил рисунок Коля.

— Вспомним,— начал я, как мы это делаем сами. Отыскиваем незанятую клетку.

— Вот и микрокалькулятору нужно будет проверять, пустует ли намеченное им место,— подхватил Сергей.

— Нельзя поставить лишние или, наоборот, не все корабли. Когда клетка найдена, мы заштрихуем ее, обозначая тем самым установленный корабль,— продолжал я.

— А в программе мы будем отмечать корабли цифрой 2,— предложил Сережа.— Рисуй Коля! (рис. 30.)

— Теперь, когда микрокалькулятор готов к игре, снова вспомним о правилах,— тоном приказа произнес Сергей.

— Противники поочередно обмениваются выстрелами — парами координат. А как объяснить микрокалькулятору, о какой клетке идет речь? — задумался Коля.

— Сергей, нельзя ли закодировать каждую клетку поля одним из двузначных чисел от 11 до 88? Первая цифра — номер строки, вторая — номер столбца,— предложил я.

— Конечно, можно. Микрокалькулятор должен проверить наличие корабля в этой клетке и, если корабль там был, стереть его обозначение и сообщить о попадании, а если корабля не было, то о промахе.

Коля уже рисовал очередной кусочек блок-схемы (рис. 31).

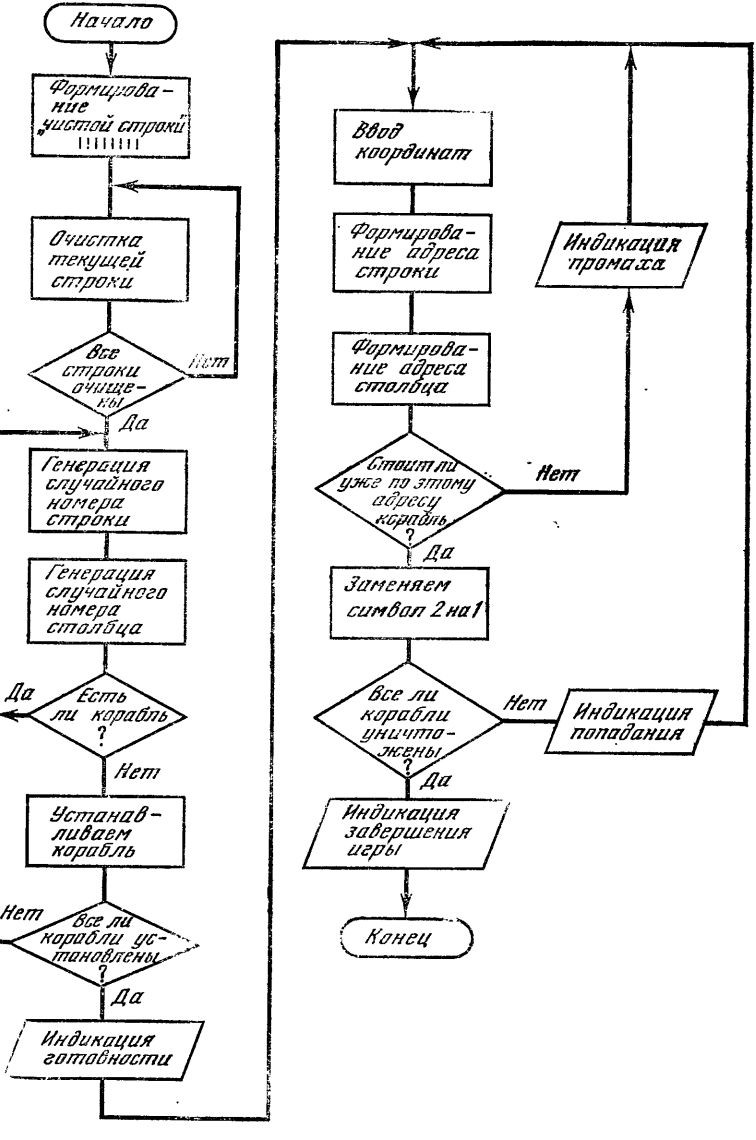

Начало
Формирование чистой строки
Очистка текущей строки
Все строки очищены
Нет
Да
Генерация случайного номера строки
Генерация случайного номера столбца
Есть ли корабль ?
Да
Нет
Устанавливаем корабль
Все ли корабли установлены ?
Нет
Да
Индикация готовности
Ввод координат
Формирование адреса строки
Формирование адреса столбца
Стоит ли уже по этому адресу корабль ?
Нет
Да
Индикация промаха
Заменяем символ 2 на 1
Все ли корабли уничтожены ?
Нет
Да
Индикация попадания
Индикация завершения игры
Конец

Рис. 32

— Давайте обозначать символом 15 готовность микрокалькулятора к игре, а символом л конец игры,— предложил я.

— И пусть «2» будет обозначать промах, а «5» попадание,— продолжил Коля.— Теперь я могу нарисовать полную блок-схему алгоритма работы микрокалькулятора (рис. 32). Сначала он расставляет корабли на своем игровом поле и останавливается, когда все корабли заняли свои места. Потом мы вводим в него координаты своего очередного выстрела. Он отвечает, попали ли мы в какой-то из его кораблей или промахнулись. Когда все его корабли уничтожены, он сообщает нам об этом.

— Ну вот, можно приниматься за программирование.— Коля гордо оглядел внушительную блок-схему.

— Чтобы не пришлось много раз менять адреса в программе, введем относительную адресацию,— предложил Сергей.

— Обычная адресация — это когда каждой команде соответствует адрес от 00 до 97,— решил уточнить я,— а вот что такое относительная адресация?

— Относительной ее называют потому, что в каждом отдельном фрагменте программы нумерация всякий раз начинается с единицы.

— Вот, скажем, первый блок в нашей блок-схеме — формирование «чистой строки», т. е. числа из восьми единиц 11111111. Проще всего записать это так.— Коля снова взялся за карандаш, и на бумаге появилось A1:1.1 2.1 3.1 4.1 5.1 6.1 7.1 8.1.— Нет, пожалуй можно и проще,— возразил он сам себе,— только как? Попробуем взять обратную величину от нашего числа. Ага, получили 9 —08. Такое число получается совсем просто — A1:1.9 2.ВП 3.8 4./—/ 5.F1/x.

— А что если представить $(9/10^8)^{-1}$ как $10^8/9$? Теперь программа займет всего четыре шага A1:1.8 2.F10^x 3.9 4.:

— Так ведь? — записал свое предложение Сергей.

— Возьмемся за блок 2. Стоп! Что это за стрелочка подходит к блоку справа снизу? — Коля заскользил карандашом по блок-схеме так, точно это была карта острова с пиратским кладом.— «Все ли строки очищены» — да ведь это же цикл!

— Коля, я забыл, что значит «цикл»,— признался я.

— Цикл — это повторяющиеся операции, которые выполняются много раз подряд до тех пор, пока одна из величин — параметр цикла — не сравняется с определен-

ным значением,— объяснил Коля.— Вот пример. Капитан приказывает: «Десять гранат по вражескому фрегату!» Число гранат — параметр цикла. Что мы должны делать? Заряжать орудие, целиться, подносить запальный фитиль и снова заряжать... до тех пор, пока не сделаем все десять выстрелов, то есть пока у нас не кончатся все десять гранат. В нашей игре параметр цикла — это число строк, а их у нас восемь. Параметр надо записать в один из регистров еще до начала выполнения цикла. Выберем для этого, например, регистр P0. Тогда фрагмент 2, фрагмент начальной записи параметра цикла — A2:1.8 2.П0.

— Вернемся к фрагменту 1,— взял в руки первый листочек Сергей.— Может быть, что-то удастся сократить. Точно! Смотрите, после выполнения фрагмента A2 на индикаторе останется число 8. А что у нас записано по первому адресу в A1? Тоже 8. Если мы поменяем A1 и A2 местами, одну из восьмерок можно будет выбросить.

На листочке появилось: A1:1.8 2.П0 и A2:1.F10^x 2.9 3.:

— Теперь в регистре X «чистая строка» 11111111, и мы можем записать ее по адресу, который хранится в P0,— листочек перешел к Коле.— Я знаю одну отличную команду — КП↑. Эта команда, как и команда КИП0, запишет содержимое регистра X по адресу, хранящемуся в P0, но в отличие от КИП0 не изменит содержимого P0. Это будет самый короткий фрагмент A3:1.КП↑. А еще я знаю, как записать «Все ли строки очищены» — A4:1.FL0 2.A3. Когда цикл закончится, мы будем уверены, что очищены все строки. Так,— вгляделся он в блок-схему,— здесь снизу опять стрелочка.

— Да, и она сливается из двух,— вмешался я.— Значит, два цикла?

— Сейчас проверим,— не отрываясь от блок-схемы, протянул Сергей,— «Есть ли корабль?» — это только проверка. Счета здесь нет, никакой параметр не изменяется. «Все ли корабли установлены?» — вот настоящий цикл. Мы будем подсчитывать число уже установленных кораблей, а когда поставим все, то пойдем дальше.

— Похоже, слова «все ли» являются признаком цикла,— предположил я.— Что же в данном случае является параметром? По-моему, это число кораблей. А сколько их будет? — обратился я к друзьям.

— Для поля 8×8 хватит 15 кораблей. Это число мы и запишем в счетчик цикла. Для образования циклов в микрокалькуляторе предназначены регистры P0, P1, P2,

ТАБЛИЦА 1

№	Команда	X	Y	РА
		4	11111111	
1	10^x	10000	11111111	
2	ПА	10000	11111111	10000
3	:	1111,1111		10000
4	1	1	1111,1111	10000
5	0	10	1111,1111	10000
6	:	111,11111		10000
7	↑	111,11111	111,11111	10000
8,9	ППВ1/11	111	111,11111	10000
10	—	1,1111—01		10000
11	1	1	1,1111—01	10000
12	0	10	1,1111—01	10000
13	×	1,1111		10000
14	2	2	1,1111	10000
15	—	—8,889—01		

ТАБЛИЦА 2

№	Команда	X	Y
13	×	2,1111	
14	2	2	2,1111
15	—	1,111—01	

Р3. В регистрах Р1, Р2, Р3 у нас хранится игровое поле, свободен только Р0, так что выбирать не из чего,— рассуждал Сережа,— А5:1.1 2.5 3.П0. Так, что там у нас дальше... «Генерация случайного номера строки». Можно взять готовую программу, ну, скажем...

Сергей повернулся к книжной полке и снял с нее тонкую книгу в черной обложке.

— Скажем, вот эту: А6:1.ИПВ 2.ВП 3.9 4.Fcos 5.Farccos 6.Fπ 7.: 8.ПВ.

— Вот здорово! Значит, номер строки у нас уже есть? — обрадовался Коля.

— К сожалению, еще нет. Эта программа «придумывает» псевдослучайные числа в диапазоне от 0 до 1, а нам нужны целые числа от 1 до 8 — адреса регистров. Надо

дополнить фрагмент 9.7 10.× 11.1 12.+,— записывал Сергей.

— Точно! Умножили на 7 — получили диапазон]0,7[, прибавили 1, диапазон стал]1,8[. Теперь по полученному адресу можем извлечь нужную строку игрового поля: 13.ПС 14.КИПС,— дополнил Коля.

— Стоп! — прервал его Сергей.— Но ведь при модификации дробная часть числа отбрасывается, а не округляется. Значит, восьмерку мы так не получим. Надо применить другой метод. Например, такой, когда к числу прибавляется 10^7 и оно сдвигается к правому краю разрядной сетки, а дробная часть выходит за этот край и отбрасывается, но самый младший разряд числа, которое остается, при этом округляется. Потом надо только из суммы вычесть 10^7, и останется округленное исходное число. Вот какие, значит, нужно выполнять команды: 13.7 14.F10^x 15.+ 16.FBx 17.— 18.ПС 19.КИПС.

Тем временем я изучал блок-схему (см. рис. 32).

— «Генерация случайного номера столбца». Но ведь столбцов столько же, сколько и строк. Значит, этот фрагмент программы будет повторяться? — размышлял я вслух.

— Нет, повторяющийся фрагмент лучше вынести в подпрограмму. Обозначим ее В1,— решил Коля и написал: В1:1.ИПВ 2.ВП 3.9 4.Fcos 5.Farccos 6.Fπ 7.: 8.ПВ 9.7 10.× 11.7 12.F10^x 13.+ 14.FBx 15.— 16.В/0, тогда фрагмент А6 будет выглядеть так: А6:1.ПП 2.В1 3.1 4.+ 5.ПС 6.КИПС, а А7 так: А7:1.ПП 2.В1.

— Представим себе, что было бы в регистрах Y и X, если бы мы на самом деле выполнили все эти команды,— предложил Сергей.

— В регистре Y — строка игрового поля, выбранная случайно, ее извлекла команда А6:6.КИПС, а в регистре X — случайный номер столбца, это благодаря подпрограмме В1,— рассуждал Коля.

— А вот как определить, есть ли уже по этому адресу корабль?

— Мы же договорились, что будем обозначать корабль цифрой 2. Вот и проверим, какая цифра стоит в выбранной строке в той позиции, номер которой указывается содержимым регистра X. Воспользуемся возможностями команды 10^x. С ее помощью мы можем получить испытательные числа 1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, если содержимое регистра X меняется от 0 до 7. У нас в регистре X число может быть любым от 1 до 8. Итак,— Сергей взял новый листок бумаги,— А8:I.F10^x 2.ПА

3.: 4.1 5.0 7.: 7.↑ 8.ПП 9.В1/11 10.– 11.1 12.0 13.× 14.2 15.–

— А что эта за странное обозначение В1/11? — не понял я.

Оно означает 11-ю команду подпрограммы В1,— объяснил Сергей и повернулся к Коле.— Теперь все ясно?

— Ясно, но не все,— напряженно изучал тот только что написанный фрагмент.— Например, адреса с седьмого по десятый — это явно отделение дробной части. Ну, а в целом...

— Попробуй проверить этот фрагмент без микрокалькулятора, «прокрутить», как говорят программисты,— посоветовал Сергей.

Коля тоже взял новый лист.— Положим, в РУ находится строка 11111111, т. е. корабля нет, а в РХ — число 4. Я буду подчеркивать тот разряд, который нас интересует (табл. 1).

Мы получили отрицательное число. А если бы корабль был 11121111, мы бы получили положительное число (табл. 2).

— Значит, нужно дополнить фрагмент 8 командами 16.Fx<0 17.А6, и блок «Есть ли корабль?» запрограммирован? — Коля вопросительно посмотрел на Сергея.

— Совершенно верно, а блок «Устанавливаем корабль» записать и вовсе легко,— Сергей добавил А9:1.ИПА 2.КИПС 3.+4.КПС.

— В самом деле,— Коля внимательно рассматривал запись,—у нас была строка 11111111 и адрес 5, тогда в этом фрагменте двойка попадет именно на 5-ю позицию: 11121111. Корабль поставлен!

— Проверкой, все ли корабли поставлены, завершается цикл,— Сергей показал мне А10:1.FL0 2.А6.

— А как мы запрограммируем блок индикации готовности к игре? — поинтересовался я.

— Число 15? А11:1.1 2.5 3.С/П,— Сергей приписал этот фрагмент к предыдущему.

— Дальше снова две стрелки,— заметил я.— Как распознать цикл, я уже знаю. Вот он: «Все ли корабли уничтожены?». Число кораблей сформировано во фрагменте А11. Наверное, можно использовать его и как параметр цикла?

— Конечно,— кивнул Коля,— но не лучше ли отправить параметр в какой-то регистр на хранение, переписать фрагмент 11 так: А11:1.1 2.5 3.П0 4.С/П? И еще вопрос: как мы будем вводить координаты выстрела?

```
AI7:I.F√ 2.C/П
AI6:I.5 2.FLO 3.AII/4
AI5:I.КИПС 2.ИПА 3.- 4.КПС
AI4:I.2 2.БП 3.AII/4
AI3:I.ПП 2.B2 3.FX<0 4.AI5
AI2:I.↑ 2.I 3.0 4.: 5.ПС 6.КИПС 7.⇄ 8.ИПС 9.- I0.I
    II.0 I2.× I3.I I4.-
AII:I.I 2.5 3.ПО 4.С/П
AI0:I.FLO 2.A6
A9:I.ИПА 2.КИПС 3.+ 4.КПС
A8:I.ПП 2.B2 3.FX<0 4.A6
A7:I.ПП 2.BI
A6:I.ПП 2.BI 3.I 4.+ 5.ПС 6.КИПС
A5:I.I 2.5 3.ПО
A4:I.FLO 2.A3
A3:I.КП↑
A2:I.FIO^X 2.9 3.:
AI:I.8 2.ПО
B2:I.FIO^X 2.ПА 3.: 4.I 5.0 6.: 7.↑ 8.ПП 9.BI/II
   I0.- II.I I2.0 I3.× I4.2 I5.- I6.B/0
BI:I.ИПВ 2.ВП 3.9 4.Fcos 5.Farccos 6.F√ 7.: 8.ПВ 9.7
   I0.× II.7 I2.FIO^X I3.+ I4.FBx I5.- I6.B/0
```

Рис. 33

Проще всего, по-моему, взять двузначные числа от 11 до 88, где цифра в разряде десятков будет означать строку, а цифра в разряде единиц — столбец. Например, 53 — значит, речь идет о пятой сверху и третьей справа клетке. Это на бумаге. А в микрокалькуляторе — пятый регистр, третий разряд.

— Начинаем программировать. A12:1.↑ 2.1 3.0 4.: 5.ПС 6.КИПС 7.⇄ 8.ИПС 9.— 10.1 11.0 12.× 13.1 14.— Понятно? — обратился Сергей к Коле.

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	8	08	32	С/П	50	64	ВП	0С
0I	ПО	40	33	↑	0Е	65	9	09
02	F 10^x	I5	34	I	0I	66	Fcos	IГ
03	9	09	35	0	00	67	Farccos	I-
04	:	I3	36	:	I3	68	F$\sqrt{}$	20
05	КП↑	LE	37	ПС	4С	69	:	I3
06	FLO	5Г	38	КИПС	ГС	70	ПВ	4L
07	05	05	39	⇄	I4	7I	7	07
08	I	0I	40	ИПС	6С	72	×	I2
09	5	05	4I	-	II	73	7	07
I0	ПО	40	42	I	0I	74	F 10^x	I5
II	ПП	53	43	0	00	75	+	I0
I2	63	63	44	×	I2	76	F Bx	0
I3	I	0I	45	I	0I	77	-	II
I4	+	I0	46	-	II	78	В/О	52
I5	ПС	4С	47	ПП	53	79	F 10^x	I5
I6	КИПС	ГС	48	79	79	80	ПА	4-
I7	ПП	53	49	F x<0	5С	8I	:	I3
I8	63	63	50	54	54	82	I	0I
I9	ПП	53	5I	2	02	83	0	00
20	79	79	52	ВП	5I	84	:	I3
2I	F x<0	5С	53	32	32	85	↑	0Е
22	II	II	54	КИПС	ГС	86	ПП	53
23	ИПА	6-	55	ИПА	6-	87	73	73
24	КИПС	ГС	56	-	II	88	-	II
25	+	I0	57	КПС	LC	89	I	0I
26	КПС	LC	58	5	05	90	0	00
27	FLO	5Г	59	FLO	5Г	9I	×	I2
28	II	II	60	32	32	92	2	02
29	I	0I	6I	F$\sqrt{}$	20	93	-	II
30	5	05	62	С/П	50	94	В/О	52
3I	ПО	40	63	ИПВ	6L			

Рис. 34

— В общем, да. Допустим, я ввел число 53; пять — номер строки, три — номер столбца. Команды 2.1 3.0 4.: дают в результате 5,3. Команды 5.ПС 6.КИПС отсекают от этого числа дробную часть, то есть дают номер строки.

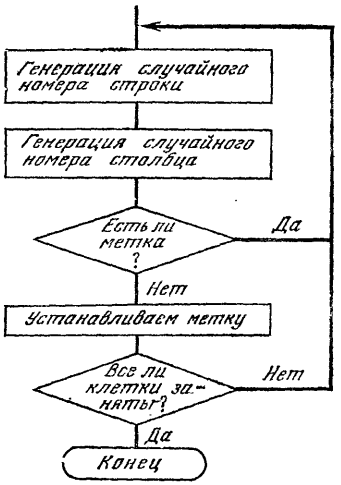


Рис. 35

	1		5	11		10	
13		6			3		
			2			9	
	7						
						8	
		12					4

1 2 3 4 5 6 7 8

Рис. 36

Рис. 37

1	6	3	8	5	2	7	4
1	2	3	4	5	6	7	8

Команда 7.⇄ на время убирает этот номер в РУ. Команды 8.ИПС 9.— дают дробную часть числа 5,3, то есть номер столбца, уменьшенный в десять раз, а умножают его на 10 команды 10.1 11.0 12.×. Так это сразу два блока! «Формирование адреса строки» и «Формирование адреса столбца». Вот здорово! Блок «Попадание» почти ничем не отличается от блока «Есть ли корабль?». Значит, повторяющиеся части этих блоков лучше выделить в подпрограмму — В2:1.F10^x 2.ПА 3.: 4.1 5.0 6.: 7.↑ 8.ПП 9.В1/11 10.— 11.1 12.0 13.× 14.2 15.— 16.В/0. Тогда блок А8 примет вид: А8:1.ПП 2.В2 3.Fx<0 4.А6, а блок А13:1.ПП 2.В2 3.Fx<0 4.А15.

— Блок индикации промаха представим тремя командами,— Сергей записал их — А14:1.2 2.БП 3.А11/4.

— Стирание корабля с точностью до знака слагаемого эквивалентно установке корабля,— внимательно рассматривал Коля исписанные листочки.— Значит, блок А15:1.КИПС 2.ИПА 3.— 4.КПС. «Все ли корабли уничтожены?» — это завершение цикла А16:1.FL0 2.А18. Если цикл завершен, то А17:1.Fл 2.С/П. А если нет? Неужели

делать отдельный блок — что-то вроде А18:1.5 2.БП 3.А11/4?

— Вспомни, как работает наш оператор цикла FL0. При всех обращениях к нулевому регистру, счетчику прохождений цикла, содержимое регистра X не изменяется. Этим мы и воспользуемся. Фрагмент А16:1.5 2.FL0 3.А11/4 заменит нам и А16, и А18, — Сергей зачеркнул ненужные команды.— Вот и все, можно переписать всю программу целиком.

Коля собрал все исписанные листки, взял еще один чистый и принялся за работу (рис. 33).

— Перенумеруем команды, заменим условные адреса на реальные, и программа готова,— Коля быстро переписал программу (рис. 34).

— Что-то мне не очень нравится наша программа,— начал Коля.— Как мы будет с ней играть? Калькулятор расставляет корабли, мы стреляем по ним. Но у нас-то самих никакого поля с кораблями нет, и калькулятор своими выстрелами ответить нам не может!

— Да, я тоже думал об этом,— кивнул Сергей.— Но вот в чем тут загвоздка: где разместить программу для ответной стрельбы? У нас остались свободными только три адреса программной памяти.

— Но у нас ведь еще один калькулятор! — не сдавался Николай.— Там и поместим программу для ответной стрельбы. Написать ее, наверное, теперь уже ничего не стоит; она будет практически повторять тот участок первой программы, где устанавливаются корабли.

— Да, только теперь нам придется устанавливать не корабли, а какие-нибудь метки, означающие, что в точку с такими координатами произведен выстрел,— продолжил я.— А проверять будем теперь не «все ли корабли установлены?», а по всем ли координатам произведен выстрел.

Коля нарисовал блок-схему (рис. 35).

— Николай, по-моему, ты забыл один важный блок. Как мы будем узнавать, по каким полям выстрелил калькулятор? — заметил я.

— Ты прав, я совсем упустил это из виду,— согласился Николай.— После того как метка установлена, надо, чтобы калькулятор сообщал нам координаты своего выстрела в виде $10A+B$, где A — строка, B — столбец.

Сергей быстро набрал программу, которую мы написали, нажал клавишу С/П и, наконец, объяснил нам, что он собирается делать.

— Боюсь,— начал он,— что наш прежний алгоритм расстановки кораблей мало подходит для расстановки меток. Их ведь 64. Представляете, сколько раз придется калькулятору «придумывать» координаты, чтобы попасть в последнюю незаполненную клетку?

— Что же делать? — огорчился Коля.

— Придется сменить алгоритм поиска свободной клетки. Есть одна идея.

Сергей нарисовал игровое поле и показал его нам (рис. 36).

— Числа в клетках — это номера меток в той последовательности, в какой они расставляются на поле. Числа с правой стороны поля — положения меток. Можете ли вы заметить какую-то закономерность в расположении чисел в клетках поля?

Мы отрицательно покачали головами.

— Здесь остался случайным лишь выбор столбца, а положение X_i, которое занимает новая метка в столбце, выбирается относительно X_{i-1}, положения предыдущей метки, например, так:

$$\text{если } X_{i-1} = \begin{cases} 1, 2, 3, & \text{то } X_i = X_{i-1} + 5, \\ 4, 5, 6, 7, 8, & \text{то } X_i = X_{i-1} - 3. \end{cases}$$

— А почему ты выбрал именно это правило для вычисления меток? — поинтересовался Коля.

— Если пользоваться этим правилом, то мы последовательно обойдем все клетки столбца. Вот, смотри,— Сергей нарисовал прямоугольник и заполнил его цифрами-ходами (рис. 37).

— А где мы будем хранить данные о предыдущей метке каждого столбца? — продолжал выяснять Коля.

— Там же, где и раньше, в регистрах Р1—Р8,— ответил Сергей.

— Положение каждой новой метки определяется положением предыдущей. Чтобы эта схема заработала, в исходный момент надо поставить в каждом столбце начальную метку,— не унимался Николай.

— Поставим,— невозмутимо отвечал Сергей.

— А как быть, если генератор выдаст номер столбца, в котором уже выбраны по очереди все восемь клеток? — Коля решил предусмотреть все варианты.

— Положение метки выражается однозначным числом, которое занимает всего один разряд каждого регистра Р1—Р8. В нашем распоряжении еще семь разрядов. Коор-

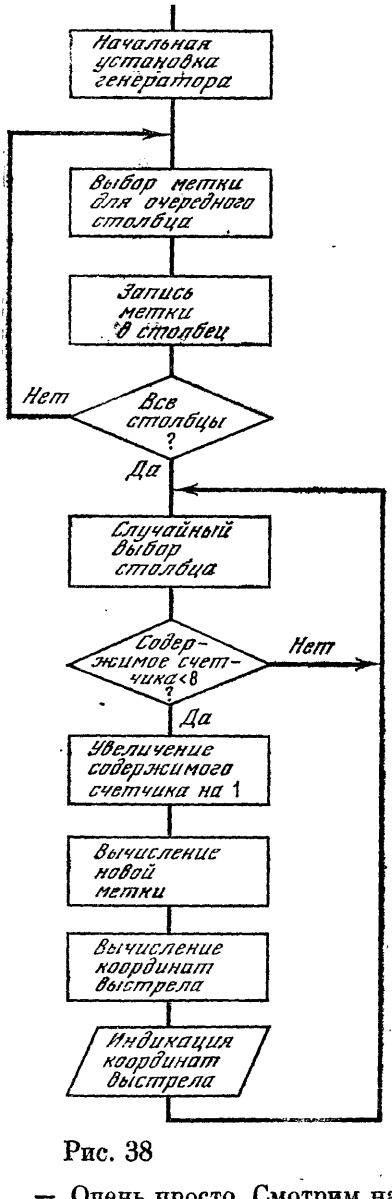


Рис. 38

динаты метки занимают младший разряд. Старший разряд можно задействовать под счетчик обращений к регистру,— пояснил Сергей.

— Тогда, раз уж мы изменили алгоритм, давайте изменим и датчик псевдослучайных чисел. Возьмем, например, такой: $\xi_k=\{5\xi_{k-1}+\pi\}$,— предложил Николай.

— Фигурными скобками ты обозначил отделение дробной части числа?— спросил я.

— Совершенно верно. Этот датчик хорош тем, что здесь все операции — быстрые. Одно число генерируется примерно за три секунды,— продолжал Коля.

Сергей тем временем начал рисовать блок-схему алгоритма (рис. 38).

— Рассмотрим блоки подробнее,— Коля потянул лист к себе.— Первый блок — это исходная установка меток. Давайте предусмотрим ввод исходного числа в регистр генератора случайных чисел с клавиатуры.

— А как его выбирать?

— Очень просто. Смотрим на часы,— сейчас 14 ч 7 мин. Пишем дробь 0,1407.

— Первый блок А1:1.ПВ, то есть перед запуском

программы достаточно набрать число 0,ЧЧММ. Оно определяется временем начала игры. На место знаков ЧЧ ставим часы, на место знаков ММ — минуты.

— Теперь, чтобы организовать счетчик, придется подготовить число-единицу в старшем разряде, то есть 10^7. Пусть оно хранится в регистре D,— предложил Сергей,— это будет блок А2:1.7 2.F10^x 3.ПД.

— Столбцов, а следовательно, меток должно быть восемь, значит, А3 может выглядеть так: А3:1.8 2.П0,— посоветовал я.

— Выбор метки для очередного столбца проще всего организовать с помощью подпрограммы,— вставил Николай,— А4:1.ПП 2.В1, той подпрограммы В1 из первой программы, она генерирует случайные числа, а записывать метку в столбец будем так — А5:КП↑.

— «Все столбцы?» запишем как А6:1.FL0 2.А4,— уверенно заметил я,— и снова обращаемся к подпрограмме,— я чувствовал, что программирование на микрокалькуляторе оказывается не таким уж сложным, как мне казалось вначале.

— Да, случайный выбор столбца проще всего организовать так: А7.1.ПП 2.В1 3.ПА 4.КИПА. А как выяснить, чему равны показания счетчика? — Коля задумался.— Рассмотрим два варианта: 10000007 и 80000007. Что их отличает? — рассуждал он.

— Попробуй вычесть из обоих чисел по $8 \cdot 10^7$,— предложил я.

— Точно, в первом случае результат оказывается отрицательным, во втором — положительным,— обрадовался Коля,— значит, $8 \cdot 10^7$ надо заранее записать в свободный регистр Р9.

— Да, и проще всего это сделать в блоке А3,— вмешался Сергей.— Тогда хватит и двух шагов А3:3.× 4.П9. Контроль содержимого счетчика запишем так: А8:1.ИП9 2.— 3.Fx<0 4.А7.

— «Увеличение содержимого счетчика на 1» — это А9:1.КИПА 2.ИПД 3.+,— я был уверен, что не ошибся. —4.↑ 5.ПС 6.ВП 7.ПС 8.3 9.— 10.F$x\neq$0 11.А9/13 12.Fx<0 13.А9/15 14.8 15.+,— Сергей дополнил мою запись и принялся пояснять.— Хотя сочетание команд ПР ВП не описано в инструкции к микрокалькулятору, но мне удалось заметить, что эти команды всегда отбрасывают старший разряд любого положительного числа. В данном случае мы отбрасываем разряд со счетчиком, т. е. в регистре X остается лишь метка. Команды с 9-й по 16-ю формируют новую метку по правилу:

```
A10:I.C/П 2.БП 3.A7
A9:I.КИПА 2.ИПД 3.+ 4.↑ 5.ПС 6.ВП 7.- 8.ПС 9.FBx
   10.3 11.- 12.Fx≠0 13.A9/16 14.Fx<0 15.A9/18 16.8
   17.+ 18.↑ 19.ИПС 20.+ 21.КПА 22.⇄ 23.1 24.0 25.×
   26.ИПА 27.+
A8:I.ИП9 2.- 3.FX<0 4.A7
A7:I.ПП 2.ВI 3.ПА 4.КИПА
A6:I.FLO 2.A4
A5:I.КП↑
A4:I.ПП 2.ВI
A3:I.8 2.ПО 3.x 4.П9
A2:I.7 2.FIO^x 3.ПД
A1:I.ПВ
```

Рис. 39

$$\text{если } X_{i-1}=\begin{cases}1, 2, 3, & \text{то } X_i = X_{i-1}+5,\\ 4, 5, 6, 7, 8, & \text{то } X_i = X_{i-1}-3.\end{cases}$$

Теперь осталось синтезировать два числа — одно вида $(n+1)000000$ м, где n — содержимое счетчика, м — новая метка, а другое число — вида 10м+ч, где ч — номер регистра. Придется чуть-чуть дополнить фрагмент A9:7.— 8.ПС 9.FBx 10.3 11.— 12.F$x\neq 0$ 13.A9/16 14.F$x<0$ 15.A9/18 16.8 17.+

Здесь команды 7.— 8.ПС выделяют содержимое счетчика, увеличенное на 1, т. е. $(n+1)0000000$. Достаточно теперь сложить его с вычисленной меткой 18.ИПС 19.+ и записать в соответствующий регистр 20.КПА. Впрочем, я поторопился, значение метки, полученное на шаге 17, еще необходимо нам для формирования числа 10м+ч. Проще всего скопировать его значение в регистр Y. Тогда 18.↑ 19.ИПС 20.+ 21.КПА. Команда 22.⇄ выводит значение м в регистр X, дальше все просто: 23.1 24.0 25.× 26.ИПА 27.+ Число 10м+ч сформировано,— закончил Сергей.

— Вывести его на индикатор поможет команда A10:1.C/П. Для нового счета 2.БП 3.A7,— Николай принялся выписывать получившиеся фрагменты (рис. 39).

— Ребята, по-моему, мы забыли о датчике псевдослу-

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ПВ	4L	22	ИПД	6Г	44	0	00
01	7	07	23	+	I0	45	×	I2
02	Г $I0^x$	I5	24	↑	0E	46	ИПА	6-
03	ПД	4Г	25	ПС	4C	47	+	I0
04	8	08	26	ВП	0C	48	С/П	50
05	П0	40	27	-	II	49	БП	5I
06	×	I2	28	ПС	4C	50	I3	I3
07	П9	49	29	Г Вх	0	5I	ИПВ	6L
08	ПП	53	30	3	03	52	5	05
09	5I	5I	3I	-	II	53	×	I2
I0	КП↑	LE	32	Г x≠0	57	54	Г π	20
II	FL0	5Г	33	36	36	55	+	I0
I2	08	08	34	Г x<0	5C	56	↑	0E
I3	ПП	53	35	38	38	57	ПВ	4L
I4	5I	5I	36	8	08	58	ВП	0C
I5	ПА	4-	37	+	I0	59	ПВ	4L
I6	КИПА	Г-	38	↑	0E	60	8	08
I7	ИП9	69	39	ИПС	6C	6I	×	I2
I8	-	II	40	+	I0	62	I	0I
I9	Г x<0	5C	4I	КПА	L-	63	+	I0
20	I3	I3	42	⇄	I4	64	В/О	52
2I	КИПА	Г-	43	I	0I			

Рис. 40

чайных чисел,— я держал в руке листочек с Колиной формулой.

— Это легко исправить,— Сергей добавил к Колиной записи еще один фрагмент:

В1:1.ИПВ 2.5 3.× 4.Fл 5.+ 6.↑ 7.ПВ 8.ВП 9.ПВ 10.8 1.× 12.1 13.+ 14.В/О

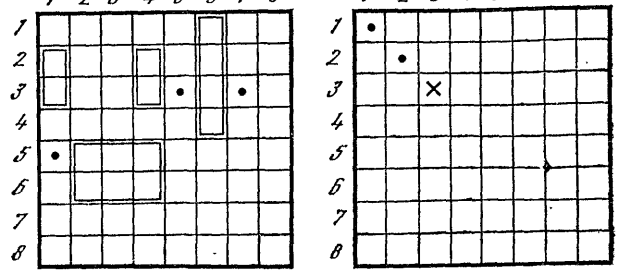


Рис. 41

Коля переписал таблицу в виде программы, заменяя относительные адреса абсолютными (рис. 40).

Сергей принялся вводить в свою «тридцатьчетверку» первую программу, Коля взял в руки более изящный МК-54. Некоторое время они нажимали на клавиши, стараясь ничего не пропустить, затем обменялись машинками, чтобы проконтролировать правильность ввода программ.

Наконец все было проверено, в регистр В первого калькулятора и в РХ второго мы ввели 0.1529 (что означало 15 ч 29 мин — стартовое время генератора псевдослучайных чисел) и запустили обе машинки.

Пока калькуляторы готовились к игре — первый «расставлял» корабли, а другой «придумывал» выстрел — мы нарисовали свое игровое поле и быстро расставили корабли — так, как на рис. 41.

Первый выстрел мы, как истинные джентльмены, отдали электронному партнеру, тем более что на индикаторе Колиного микрокалькулятора уже высвечивалось 37.

— Мимо! — хором воскликнули мы и набрали на клавиатуре Сережиной машинки ответный ход 11.

Калькулятор несколько раз «моргнул» и высветил «2» — промах. Следующий ход «МК-54» был 51 — тоже промах. Ответный выстрел 22 — снова «2». 35 — мимо, 33 — «5»!

— Ура!

Первое попадание ободрило нас, еще бы — можно сделать дополнительный выстрел...

Здесь мы оставим ребят. Сообщим лишь результат поединка — выиграла «Электроника».

Для тех, кто решился попробовать свои силы в игре с машинками, приведем инструкцию к игре.

1. Запишите программу 1 в калькулятор 1 и программу 2 в калькулятор 2.

2. Введите число 0, ЧЧММ в РВ калькулятора 1 и в РХ калькулятора 2, запустите обе машины на счет клавишами С/П. (Напомним, что число 0, ЧЧММ получают, взглянув на часы: ЧЧ — часы, ММ — минуты. Например, число 0,2045 получено из 20 ч 45 мин).

3. Пока калькуляторы считают, нарисуйте игровое поле и расставьте свои корабли (15 кораблей).

4. На индикаторе калькулятора 2 — очередной выстрел электронного партнера в виде AB, где A — строка, B — столбец. Независимо от результата выстрела нажмите клавишу С/П на панели калькулятора 2, чтобы он заблаговременно подготовил свой следующий выстрел.

5. Если выстрел калькулятора 2 попал в цель, то переходите к п. 4. В противном случае переходите к п. 6.

6. Наберите на клавиатуре калькулятора 1 свой выстрел также в виде АВ и нажмите на клавишу С/П.

7. Если на индикаторе калькулятора 1 загорелась «2» — вы промахнулись, переходите к п. 4. В противном случае, если на индикаторе — «5», отметьте уничтоженный корабль противника и переходите к п. 6. Помните, что при этом вы располагаете правом на дополнительный выстрел. Если на индикаторе калькулятора 1 появилось число л, это означает, что вы одержали победу. Игра закончена.

Глава 6

„ЧТО НАША ЖИЗНЬ? — ИГРА!"

Заключительная глава, последняя из ступенек пройденного в книге пути к искусству программирования на микрокалькуляторе. С достигнутой высоты за играми, разбираемыми в этой главе, читателю станут видны понятия различных разделов математики, и прежде всего теории игр, дисциплины вполне серьезной, находящей все более широкое практическое приложение в экономике, управлении, прогнозировании будущего.

«Что наша жизнь? — Игра!» Удивительно, насколько точно это высказывание с точки зрения математика. Ибо и наша повседневная жизнь, и игра построены на постоянном разрешении конфликтов.

Конфликт — это ситуация, при которой разные участники имеют разные, несовпадающие цели. Именно так определяет это понятие теория игр — специальная математическая дисциплина, занимающаяся игровыми проблемами.

Конфликты возникают в нашей жизни постоянно. Причем не только житейские — между отцами и детьми, физиками и лириками, но и в более широком смысле — между человеком и природой.

Допустим, перед агрономом стоит проблема: какой культурой засевать поле? Культура А дает более высокий урожай, но очень чувствительна к погоде. Культура Б плодоносит меньше, зато почти не реагирует на засуху и проливные дожди.

Да простят нас агрономы, но с точки зрения теории игр эта проблема не отличается от проблемы карточного игрока, который, не зная карт партнера, размышляет: как ему пойти лучше? Так, чтобы при благоприятном раскладе получить больший выигрыш, но с риском много проиграть? Либо на небольшой выигрыш с малым риском?

Разница только в том, что в карточной игре партнер в ответ на ход игрока скорее всего сделает самый сильный ход, а в предыдущем случае партнер агронома — природа, наверное, ответит случайным образом.

В обоих случаях налицо конфликт. Его исследование, выработка оптимальной стратегии как раз и составляют основную проблему теории игр. Под стратегией в этой теории понимается набор правил, которые определяют, какой из имеющихся ходов необходимо сделать в сложившейся игровой ситуации для получения наибольшего выигрыша (или наименьшего проигрыша) в игре.

Днем рождения теории игр считают 29 июля 1654 г. Именно в тот день Б. Паскаль отправил П. Ферма письмо, в котором затронул игровые проблемы. Но в научную дисциплину эта теория сложилась только в нашем веке. В 1927 г. французский математик Э. Борель сформулировал фундаментальную теорему теории игр о существовании оптимальных стратегий. Доказал ее в следующем году американский математик Дж. фон Нейман, опубликовавший работу «К теории стратегических игр». Наконец, он же в соавторстве с О. Моргенштерном в 1944 г. опубликовал книгу «Теория игр и экономическое поведение», которая теперь по праву считается основополагающей работой в теории игр.

Но особых успехов новая теория добилась в наше время, когда были созданы ЭВМ и появилась возможность моделировать с их помощью довольно сложные игровые ситуации.

Кстати, мысль использовать ЭВМ для программирования игр возникла на самой заре компьютерной эры. Один из основоположников кибернетики К. Шеннон в 50-х годах написал статью «Играющие машины». Там он предложил делить машины, способные выполнять роль партнеров в играх, на три категории. Самый простой тип — ЭВМ, способные играть в полностью определенные (детерминированные) игры, когда точно известна стратегия, позволяющая делать выигрышный ход в каждой игровой ситуации. К таким играм можно отнести «крестики-нолики» и всевозможные модификации игры «ним» (об одной ее модификации — «игре на дорожке» — мы еще поговорим подробнее).

Машины второго типа могут участвовать в играх, полный анализ которых хотя и допустим теоретически, но практически невозможен в силу их сложности и для которых по этой причине ясны лишь самые общие принципы построения правильной стратегии (шашки, шахматы, многие карточные игры).

Третий и, по мнению Шеннона, самый сложный тип играющих машин — это ЭВМ, способные улучшать свою игру на основе полученного опыта.

Современная классификация игр сложна, содержит множество терминов, для пояснения которых понадобилось бы немало места. Не будем углубляться в эти сложности. Для понимания основных идей теории игр нам достаточно будет классификации Шеннона с ее тремя разделами.

Начнем с игр первого типа. Их характерная особенность в том, что по крайней мере для одного из партнеров существует строго определенная стратегия, следуя которой он независимо от ходов соперника никогда не проиграет. Поэтому при программировании подобных игр первоочередная задача — построить выигрышный алгоритм. Если игра достаточно проста, то играть в нее по этому алгоритму сможет даже микрокалькулятор. Нам останется лишь написать для него соответствующую программу.

Рассмотрим, как это делается, на конкретном примере «Игры на дорожке». Суть ее в том, что два противника двигают фишки, находящиеся на противоположных сто-

ронах дорожки, разбитой на n полей, например, на одной из вертикалей шахматной доски, где дорожка содержит восемь полей. Ходы делаются поочередно. За один ход каждый участник может подвинуть свою фишку не более чем на m полей вперед или назад. При этом нельзя перепрыгивать через фишку партнера и покидать пределы

Рис. 42

дорожки. Проигрывает тот, у кого не окажется ходов, т. е. чья фишка будет загнана в одно из двух крайних полей дорожки (рис. 42).

Прежде всего попытаемся определить конечную ситуацию, гарантирующую выигрыш. Она довольно очевидна. Если расстояние между фишками перед ходом партнера равно $(m+1)$ полю, то, какой бы ход он ни сделал, мы своим следующим ходом поставим свою фишку вплотную к его фишке, и он принужден будет отступать. Теперь выигрыш — дело техники. Когда партнер начнет пятиться назад, мы просто будем копировать его ходы, наступая, и таким образом неминуемо загоним его в крайнее поле.

Итак, надо добиться, чтобы перед последним ходом партнера наши фишки разделяло $(m+1)$ поле. Анализируя «назад», мы придем к выводу, что перед предпоследним ходом партнера расстояние между фишками должно составлять $2(m+1)$ поле. Продолжая рассуждать, придем к простому положению: перед первым ходом партнера фишки должна разделять дистанция, измеряемая $l(m+1)$ полем, где l — любое целое число. Тогда независимо от того, как будет ходить партнер, мы всегда будем иметь возможность сократить к его очередному ходу расстояние между фишками до целого числа отрезков длиной $(m+1)$.

Так как наши интересы в игре будет представлять калькулятор, то нам как тренерам необходимо обучить выигрышной стратегии именно его. Иначе говоря, надо составить программу, играя по которой он всегда бы выигрывал.

Для начала наш подопечный должен будет проанализировать, делится ли число $(n-2)$, выражающее исходное расстояние между фишками, на число $(m+1)$ без остатка. Если не делится, следует определить оста-

ток от этого деления, т. е. число $x_0=(n-2)-(m+1)\times$ $\times[(n-2)/(m+1)]$ (символом [ ] обозначена целая часть заключенной в такие скобки величины). На столько полей и должен шагнуть вперед калькулятор своим первым ходом. А если делится? В этом случае — одно из двух. Либо калькулятор из уважения к сопернику должен будет сказать: «Сдаюсь» (ведь соперник скорее всего сильный, и, наверное, выигрышный алгоритм для него не секрет). Либо из еще большего уважения — предоставить почетное право первого хода сопернику, а затем скромно перейти к абсолютно выигрышной стратегии. Оставим на время этические соображения в стороне, выберем для нашего «спортсмена» второй путь.

Далее. Каждый следующий ход микрокалькулятора x_i определяется по простой формуле:

$$x_i=\begin{cases} m+1-y_i, & \text{если } y_i>0, \\ -y_i, & \text{если } y_i<0, \end{cases}$$

где y_i — i-й ход партнера. Таким образом, во время всей игры будет сохраняться инвариантным (неизменным) соотношение: расстояние между фишками $p=l(m+1)$, где l — любое целое число.

Если оценить эту игру с позиций кибернетики, то ситуацию, когда выполняется приведенное выше соотношение, следует считать равновесной. Каждым своим ходом противник нарушает равновесие, а ответным ходом калькулятор его восстанавливает. Алгоритм, как говорят, является следящей системой с обратной связью.

Стратегия, которую мы описали, гарантирует нашему подопечному выигрыш при игре с любым партнером, пользующимся любой стратегией. Реализуем ее теперь в виде программы.

Правда, остался невыясненным один вопрос: «А судьи кто?» В каждой игре должен быть судья, следящий за соблюдением правил и фиксирующий победу.

Что касается микрокалькулятора, то, обученный играть по нашему алгоритму, он никогда не попытается перепрыгнуть через фишку партнера или сделать ход на большее число клеток, чем предусмотрено правилами. А вот партнер калькулятора, человек, даже если мы нисколько не сомневаемся в его честности, может сделать неправильный ход просто по ошибке. Так как специального судьи в подобной игре, по всей вероятности, не будет, судейские обязанности следует возложить на каль-

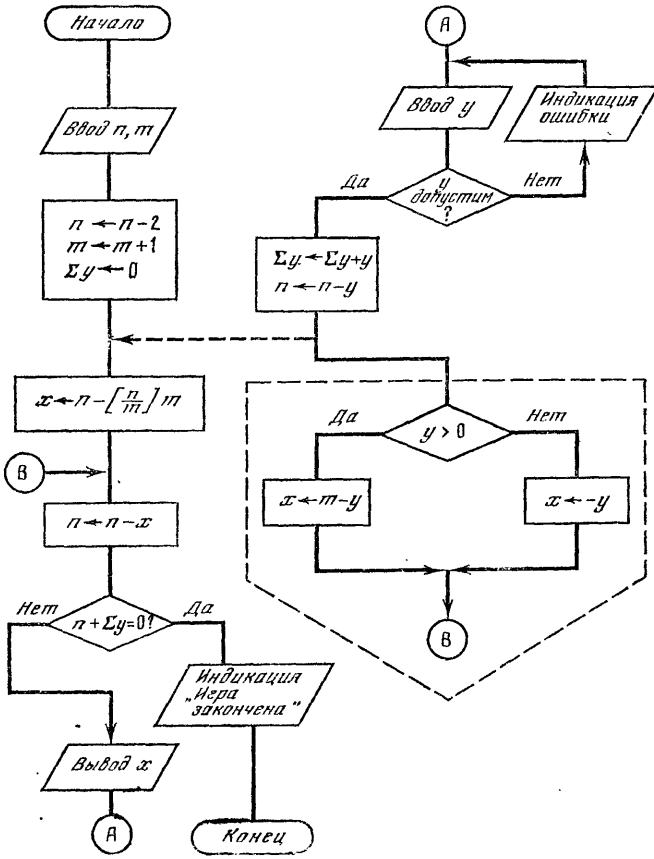


Рис. 43

кулятор — пусть он проверяет правильность ходов партнера и указывает на допущенные ошибки. И пусть он же фиксирует победу (наверное, эта функция будет для него особенно приятна: ведь победителем всегда будет он).

Блок-схема алгоритма «Игры на дорожке», учитывающая все эти соображения, показана на рис. 43. Программа, составленная по этой блок-схеме С. И. Шапиро, приведена на рис. 44.

Алгоритм можно представить в более компактной форме. Для этого надо удалить последнее ветвление (на схеме обведено пунктиром) и передавать управление на

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	↑	0E	24	−	11	48	ИП1	61
01	0	00	25	ПД	4Г	49	ИПС	6С
02	ПС	4С	26	ИПС	6С	50	+	10
03	ГО	25	27	+	10	51	Гx<0	5С
04	2	02	28	Гx=0	5Е	52	57	57
05	−	11	29	32	32	53	КЗ	30
06	ПД	4Г	30	Гπ	20	54	КНОП	54
07	⇄	14	31	С/П	50	55	БП	51
08	1	01	32	ИПВ	6L	56	34	34
09	+	10	33	С/П	50	57	ПС	4С
10	ПА	4−	34	П1	41	58	ИПД	6Г
11	:	13	35	Г x≠0	57	59	ИП1	61
12	1	01	36	53	53	60	−	11
13	+	10	37	Г x^2	22	61	ПД	4Г
14	ПО	40	38	Г$\sqrt{}$	21	62	ГВх	0
15	КИПО	Г0	39	ИПА	6−	63	/−/	0L
16	ИПД	6Г	40	−	11	64	Гx<0	5С
17	ИП0	60	41	Гx<0	5С	65	21	21
18	ИПА	6−	42	53	53	66	ИПА	6−
19	×	12	43	ИП1	61	67	+	10
20	−	11	44	ИПД	6Г	68	БП	51
21	ПВ	4L	45	−	11	69	21	21
22	ИПД	6Г	46	Гx<0	5С			
23	⇄	14	47	53	53			

Рис. 44

блок, где вычисляется первый ход (также показано пунктиром). Иначе говоря, следует каждый раз сводить задачу к предыдущей с новым значением n. Программа при этом станет короче на несколько команд, зато увеличится время ее работы. Для расчета каждого хода электронному партнеру придется выполнять на несколько

вычислительных операций больше. А так как наша миниатюрная ЭВМ не сверхбыстродействующая, то на времени игры предпринятое усовершенствование алгоритма отразится довольно заметно. Ради экономии нескольких секунд на обдумывание каждого хода лучше, пожалуй, оставить прежний, несовершенный алгоритм.

Работать, а точнее, играть с микрокалькулятором по приведенной программе просто. После ее ввода и перехода в режим вычислений нужно ввести параметры игры: $m\uparrow n$ В/О С/П. После некоторого раздумья калькулятор высветит свой первый ход или «нуль», что означает передачу почетного права первого хода вам. В любом случае калькулятор готов к вашему ходу. **Наберите желаемое число на клавиатуре и нажмите С/П.** Если ваш ход противоречит правилам игры, то на индикаторе появится сообщение ЕГГОГ. Ход надо повторить. При желании вы можете проконтролировать **расстояние между фишками**, нажав клавиши ИП Д, и расстояние, на которое вы **сдвинулись** от **начальной позиции**, нажав клавиши ИП С.

В конце игры, когда микрокалькулятор выиграет (к сожалению для вас, это будет всегда), на индикаторе появятся 8 цифр числа π.

Можете приглашать своих друзей и предлагать им сразиться с вашим подопечным. Только скорее всего это им быстро наскучит. Что за интерес играть, когда у тебя нет никакой надежды выиграть? Ведь даже школьник, садящийся играть в шахматы с гроссмейстером, имеет отличные от нуля шансы на победу. Здесь же вероятность вашего выигрыша строго равна нулю.

Такого рода игры имеют чисто теоретическое значение. В жизни их аналогов почти не встречается. Значительно чаще приходится сталкиваться с ситуациями, когда неизвестны предшествующие действия партнера и приходится оценивать вероятность каждого из его последующих ходов. В роли партнера может выступать не только противник, скажем, по игре в шахматы, но и настоящий противник в настоящей войне и, что бывает сплошь и рядом, природа при промышленных или сельскохозяйственных «играх».

Для анализа подобных игр обычно используют таблицы (матрицы) с результатами сочетания различных стратегий противников. Поэтому такие игры называют еще матричными.

Если в игре два участника, то матрица будет состоять из строк (это возможные стратегии первого участника) и столбцов (стратегии второго участника). Заметим, что под стратегией в простейшем случае следует понимать ход. В каждой клетке матрицы на пересечении i-й строки и j-го столбца записывают число c_{ij} — выигрыш 1-го игрока в случае, если он выбирает i-ю стратегию, а его противник выбирает j-ю. Эта величина будет положительной, если игрок действительно выигрывает, и отрицательной при его проигрыше (выигрыше со знаком «минус»).

Рассмотрим для примера весьма простую игру, описанную М. Гарднером в книге «Математические новеллы». Игроки A и B выбрасывают одновременно один или два пальца, после чего B уплачивает A столько долларов, сколько пальцев выбросили они оба. Кстати, заметьте: в отличие от ранее рассмотренных игр ходят партнеры одновременно, а не по очереди. Это очень существенно.

Матрица для этой игры с точки зрения игрока A будет такой:

A/B	1	2
1	2	3
2	3	4

В клетках матрицы указаны выигрыши игрока A при различных сочетаниях доходов (стратегий) партнеров. Минимальный его выигрыш, т. е. выигрыш при наихудшем для него ходе партнера, равен двум по 1-й строке и трем по 2-й строке матрицы. Максимальная из этих величин, тройка, называется максмином. Еще ее называют нижней ценой игры. Она равна выигрышу игрока A при его наилучшей стратегии и наихудшей (с его точки зрения) стратегии игрока B. В нашей матрице максмин находится в первой клетке 2-й строки.

Для второго игрока максимальный проигрыш равен либо четырем (по 2-му столбцу), либо трем (по 1-му). Минимальная из этих величин называется минимаксом. Он равен трем. Это верхняя цена игры. Эта величина характеризует минимум проигрыша игрока B при наилучшей его стратегии и наихудшем для него ходе про-

тивника. Находится минимакс во 2-й клетке 1-го столбца.

Если, как в нашем примере, положения минимакса и максмина совпадают, то говорят, что игра имеет седловую точку. Она как раз и находится в общей для минимакса и максмина клетке матрицы. При этом существуют оптимальные стратегии для обоих игроков. Для игрока A это строка, где находится седловая точка, а для игрока B — столбец, содержащий эту точку. Выбирая в соответствии с таблицей свою стратегию, игрок A максимизирует свой выигрыш, а игрок B минимизирует свой проигрыш.

Величина, находящаяся в седловой точке, называется просто «ценой игры». Видно, что при выборе оптимальной стратегии выигрыш первого игрока не меньше цены игры, а проигрыш второго не больше цены игры. Игра называется справедливой, если ее цена равна нулю, т. е. когда оба соперника имеют одинаковые шансы на победу. Игра Гарднера явно несправедлива. Выиграть может только игрок A.

Наша игра позволяет пояснить еще один термин теории игры — чистая стратегия. Так называется стратегия, которой нужно придерживаться постоянно. Иначе говоря, первый игрок всегда должен выбрасывать два пальца, а второй — один.

Предвидим недоумения некоторых читателей. Зачем эти сложности — матрицы, седловые точки? Самые элементарные соображения, построенные с помощью одного лишь здравого смысла, без всяких таблиц и терминов, позволили бы сделать те же выводы.

Что ж, если теория совпадает со здравым смыслом, то хвала... здравому смыслу.

Усложним нашу игру. По-прежнему участники могут выбрасывать один или два пальца, но первый игрок будет считаться победителем только в том случае, если сумма пальцев четная, иначе победу будет праздновать его соперник.

Вот матрица этой игры:

A/B	1	2
1	1	−1
2	−1	1

Здесь максмин первого игрока равен минус единице, а минимакс второго — единице. Они не совпадают, значит, седловой точки не существует. Потому, хоть мы и знаем верхнюю и нижнюю цену игры, просто цена этой игры нам неизвестна. Мы, следовательно, не можем утверждать даже, справедлива ли игра.

Вопрос об определении цены игры оставим пока открытым. Пока же скажем только, что не следует путать цену игры со стоимостью игры. Стоимостью называется сумма выигрышей и проигрышей партнеров. Поэтому в игре двух соперников стоимость всегда равна нулю — сколько один выиграл, столько другой проиграл.

Какую же стратегию предложит выбрать в этой игре «здравый смысл»? Все время выбрасывать один палец? Но соперник-то наш тоже не лыком шит. Быстро разобравшись в этой стратегии, он начнет раз за разом выигрывать. То же самое можно ждать от него и при любой другой строго определенной стратегии, скажем, 1, 2, 1, 2... или 1, 1, 2, 1, 1, 2 и т. д.

Порок таких стратегий именно в их организованности.

Послушаем теперь, что скажет теория. Согласно ей единственно приемлемым в таких играх является абсолютно случайный набор различных стратегий. Только такой стратегической коктейль может обеспечить нулевой проигрыш в этой игре независимо от стратегии партнера. Дело в том, что случай имеет то преимущество перед любым волевым актом, что не подчиняется никаким закономерностям. Иначе говоря, сколько ни анализируй последовательность ходов Случая, предугадать следующий ход невозможно.

Итак, отсутствие седловой точки и отсутствие чистой выигрышной стратегии. Нет ли связи между этими признаками игры? Есть: из первого вытекает второй. Правда, не следует думать, что отсутствие седловой точки не позволяет выбрать оптимальную стратегию. Позволяет — только в этом случае стратегия всегда будет смешанной, будет представлять собой смесь из некоторых стратегий.

Рассмотрим еще одну игру. Ее предложил американский математик Р. Айзекс. Состоит она в следующем. У каждого из двух играющих имеются по две карты, склеенные рубашками. У одного — туз черной масти и восьмерка красной, у другого — красная двойка и черная семерка. Игроки одновременно выкладывают свои двусторонние карты. Если цвета карт совпали, выигрывает

первый игрок, в противном случае — второй. Сумма выигрыша во всех случаях равна числу фигур на карте победителя.

Матрица игры Айзекса имеет такой вид:

А/В	*7ч*	*2к*
1ч	*1*	*-2*
8к	*-7*	*8*

Седловой точки в этой матрице нет. Какую же стратегию выбрать, чтобы выиграть (или хотя бы не проиграть)? И вообще, справедлива ли эта игра, т. е. оба ли партнера имеют равные шансы на выигрыш? Здравый смысл подсказывает, что да, ведь сумма выигрышей первого игрока (положительные числа) в точности равна сумме выигрышей второго (отрицательные числа). А теория?

Если матрица игры не имеет седловой точки, то для достижения успеха в такой игре необходимо выбирать стратегии, смешанные в определенной пропорции. Покажем, как определять их.

Обозначим частоту выбора первой стратегии для игрока A символом p_1, частоту выбора второй стратегии — p_2. Те же величины для игрока B обозначим q_1 и q_2. Тогда стратегию первого игрока S_A и стратегию второго игрока S_B можно записать так:

$$S_A = \begin{vmatrix} A_1 & A_2 \\ p_1 & p_2 \end{vmatrix}, \qquad S_B = \begin{vmatrix} B_1 & B_2 \\ q_1 & q_2 \end{vmatrix}.$$

Задача состоит в определении величин p_i и q_j — частот или вероятностей использования разных стратегий.

Пара связывающих их соотношений очевидна:

$$\sum_{i=1}^{n} p_i = 1, \qquad \sum_{j=1}^{m} q_i = 1$$

($i=1, 2, \ldots, n$; $j=1, 2, \ldots, m$; n — число строк, m — число столбцов матрицы игры c_{ij}). Они вытекают из определения искомых величин как вероятностей: сумма вероятностей должна быть равна единице. Другие соотношения определяются из следующих соображений. Средний вы-

игрыш стороны А при выборе смешанных стратегий равен

$$Y=\sum_{i=1}^{n}\sum_{j=1}^{m}c_{ij}p_iq_j=c_{11}p_1q_1+c_{12}p_1q_2+c_{21}p_2q_1+c_{22}p_2q_2.$$

Для определения значения переменных, при которых величина Y достигает максимума, нужно приравнять нулю ее частные производные по p_i и q_j:

$$\partial Y/\partial p_i=0, \quad \partial Y/\partial q_j=0.$$

Подставив в эти формулы значение Y и продифференцировав, мы получим дополнительные соотношения, связывающие искомые переменные. Нетрудно видеть, что их будет даже больше, чем переменных. В нашем примере получится 6 уравнений для определения 4 неизвестных. От «лишней» пары уравнений можно освободиться так. Выражают p_n и q_m через их предыдущих собратьев (скажем, p_2 и q_2 через p_1 и q_1) и подставляют в формулу для Y. Теперь уравнений столько, сколько нужно.

Поступим и мы так же. Обозначим $p=p_1$ и $q=q_1$. Подставив $p_2=1-p$ и $q_2=1-q$ в уравнение для Y и продифференцировав, получим два уравнения для определения p и q.

Чтобы не утомлять читателя промежуточными выкладками, приведем сразу конечный результат:

$$p=\frac{c_{22}-c_{21}}{c_{11}-c_{12}-c_{21}+c_{22}}; \qquad q=\frac{c_{22}-c_{12}}{c_{11}-c_{12}-c_{21}+c_{22}}.$$

Нужно заметить, что вероятность выражается положительным числом, не превосходящим единицы. Поэтому если в результате решения какие-либо значения величин p_i и q_j получатся отрицательными, то соответствующие вероятности надо приравнять нулю, если же они будут больше единицы, то ограничить их значения единицами.

Вернемся к формуле для определения выигрыша игрока A. Подставив в нее полученные значения вероятностей, т. е. частот стратегии, получим для максимального выигрыша формулу

$$Y_{\max}=\frac{c_{11}c_{22}-c_{12}c_{21}}{c_{11}-c_{12}-c_{21}+c_{22}}.$$

Заменив в этой формуле символы соответствующими числами из нашей матрицы, мы получим неожиданный результат. Оказывается, при самой оптимальной стратегии

выигрыш игрока A равен... минус одной трети. Иными словами, в самом лучшем случае игрок A может рассчитывать на проигрыш в среднем одного доллара в трех партиях.

Вот вам и «здравый смысл»!

Кстати, приведенная формула позволяет объективно оценить справедливость игры. Если числитель ее равен нулю, т. е. выполняется соотношение $c_{11}/c_{12}=c_{21}/c_{22}$, игра справедлива. Это значит, что средний выигрыш каждой из сторон при выборе оптимальной стратегии будет равен нулю. Во всех остальных случаях кто-либо из партнеров может оказаться внакладе при выборе любого стратегического коктейля.

Какова, кстати, оптимальная стратегия игрока A в игре Айзекса? Она состоит в смеси стратегии 1 и стратегии 2 в пропорции 5/6:1/6, т. е. 5:1. Иначе говоря, туз нужно показывать в среднем в 5 раз чаще, чем восьмерку. Разумеется, выбор этот должен быть абсолютно случаен, иначе партнер может приспособиться к чередованию стратегий, и тогда одним долларом в каждых трех партиях не отделаться.

Если вы хотите играть в такую игру, то советуем вооружиться игральным кубиком и перед каждым ходом подбрасывать его. Выпала, скажем, единица, выбираем вторую стратегию, во всех остальных случаях — первую.

Если же кубика под руками не оказалось, а микрокалькулятор свободен от рабочих расчетов, то он с успехом может играть роль помощника в этой игре.

Введите в него программу, приведенную на рис. 45. После запуска программы (В/О, С/П) через 15—20 с нажмите клавишу С/П, и вы прочтете результат первого бросания кубика. Затем надо только нажимать С/П и, глядя на индикатор, делать ход. Может статься, что на индикаторе будет «0». Тогда считайте, что кубик покинул игровое поле, и его надо перебросить, т. е. запустить программу вновь.

Вернемся еще раз к играм, описанным в начале главы («игра на дорожке», «ним») и им подобным. Их принципиальное отличие от игр Гарднера и Айзекса в том, что они полностью определены. Делая ход, мы всегда знаем предыдущий ход партнера. Теория говорит, что матрица таких игр всегда имеет седловую точку. А потому для них существует чистая стратегия, гарантирующая наибольший успех. Кстати, к таким играм относятся и шахматы. Теоретически и для них можно со-

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	↑	0E	06	↑	0E	I2	↑	0E
0I	6	06	07	2	02	I3	Сх	0Г
02	↑	0E	08	↑	0E	I4	БП	5I
03	I	0I	09	3	03	I5	00	00
04	↑	0E	I0	↑	0E			
05	.4	04	II	5	05			

Рис. 45

ставить полную матрицу, перебрав на каждый ход партнера все возможные ответные ходы, и таким образом найти стратегию, следуя которой мы никогда не проиграли бы даже чемпиону мира. К счастью (для шахмат), матрица эта настолько огромна, что вряд ли может быть «расписана» даже в обозримом будущем. Потому-то шахматы в отличие, например, от «крестиков-ноликов» и пользуются столь большой популярностью.

Игры, где неизвестны предыдущие ходы партнеров, имеют седловую точку не всегда. По этому поводу мы уже говорили: если седловой точки нет, то надобно выбирать смесь стратегий, варьируя их наборы с определенной частотой, но абсолютно случайным образом. Только это гарантирует секретность стратегических планов и невозможность для соперника приспособиться к ним.

Впрочем, почему мы все время говорим, что соперник может разгадать нашу стратегию? А не попытаться ли нам разгадать стратегию соперника? Только в этом случае можно добиться выигрыша в играх с нулевой ценой.

Здесь мы вплотную подходим к третьему, по классификации К. Шеннона, типу играющих машин или, лучше сказать, играющих программ. Речь идет о самообучающихся программах.

В чем состоит процесс обучения? В конечном счете — в накапливании знаний и умений на их основе делать целесообразные выводы. А это под силу практически любой ЭВМ, в том числе и нашему калькулятору.

В свое время К. Шеннон спроектировал несложный автомат для игры в чет-нечет. Его машина анализировала ходы партнера-человека и пыталась определить закономерности в их следовании. Это единственный путь

Адрес	Команда	Код	Адрес	Команда	Код	Адрес	Команда	Код
00	ИП8	68	33	ИП2	62	66	-	II
0I	П9	49	34	+	I0	67	К x≠0 В	7L
02	ИП7	67	35	2	02	68	I	0I
03	П8	48	36	×	I2	69	-	II
04	+	I0	37	ИП3	63	70	К x≠0 В	7L
05	ИП3	63	38	+	I0	7I	КИП4	Г4
06	П7	47	39	2	02	72	ИПС	6C
07	+	I0	40	×	I2	73	ВП	0C
08	ИП2	62	4I	ИП7	67	74	9	09
09	П3	43	42	+	I0	75	Fcos	IГ
I0	+	I0	43	К x≠0 А	7-	76	Farccos	I-
II	ИПI	6I	44	I	0I	77	F π	20
I2	П2	42	45	-	II	78	:	I3
I3	+	I0	46	К x≠0 А	7-	79	ПС	4C
I4	ИП0	60	47	2	02	80	ИПД	6Г
I5	ПI	4I	48	-	II	8I	-	II
I6	+	I0	49	К x≠0 В	7L	82	К x<0 В	CL
I7	6	06	50	2	02	83	ИП0	60
I8	:	I3	5I	-	II	84	F x=0	5E
I9	ПД	0Г	52	К x≠0 В	7L	85	00	00
20	ИП6	60	53	I	0I	86	БП	5I
2I	ИП5	65	54	-	II	87	9I	9I
22	-	II	55	К x≠0 А	7-	88	ИП0	60
23	ИП5	65	56	3	03	89	F x≠0	57
24	С/П	50	57	-	II	90	00	00
25	П0	40	58	К x≠0 В	7L	9I	КИП5	Г5
26	КИП6	Г6	59	I	0I	92	Cx	0Г
27	ИП4	64	60	-	II	93	П4	44
28	F x=0	5E	6I	К x≠0 А	7-	94	БП	5I
29	7I	7I	62	2	02	95	00	00
30	ИПI	6I	63	-	II			
3I	2	02	64	К x≠0 А	7-			
32	×	I2	65	2	02			

Рис. 46

к победе. Любопытно, что коллега К. Шеннона Д. Хагельберджер создал подобный же автомат. Потом была создана еще одна машина, передающая ходы автоматов друг другу, и появилась возможность устраивать «межмашинные матчи». Как с гордостью сообщает К. Шеннон, его машина выигрывала чаще. К сожалению, он не приводит алгоритма, которым пользовался его автомат для выбора стратегий. Суть его тем не менее понятна. Нужно проанализировать предыдущие ходы партнера и на основе выявленной закономерности сделать вывод о наиболее вероятном следующем ходе.

На этом же принципе основана программа для микрокалькулятора, позволяющая ему выступать в роли «думающего» партнера (рис. 46).

Программа получилась довольно большая. Еще бы, моделировали мы ни много ни мало — мышление. Занята практически вся программная память нашего «мыслителя», поэтому удобством работы с программой пришлось пожертвовать — некоторые предварительные операции нужно будет сделать вручную. Перед запуском калькулятора надо очистить регистры 2, 5, 6, 7, 9 и С, занести единицу в регистры 0, 1, 3 и 8 и ввести число 83 в регистр А, число 88 — в регистр В. Затем нужно нажать клавиши В/О, С/П. Через несколько секунд на индикаторе появится «0». Калькулятор готов к сражению. Теперь ваша задача проста: после каждого останова вводить цифру 0 или 1 и нажимать С/П. Число, появляющееся на индикаторе во время игры,— это число побед калькулятора, т. е. число случаев, когда задуманное вами число отгадано. Число ваших побед — в регистре Y. Нажимая клавишу $\rightleftarrows$, вы можете узнавать общий счет и пытаться контролировать ход поединка.

Чтобы получить объективное представление о «мыслительных способностях» вашей «Электроники БЗ-34», советуем сыграть с ней десяток-другой партий до чтения следующих страниц — там будет описан алгоритм ее работы, зная который вы скорее всего сможете доказать свое преимущество перед бездушным автоматом. Кроме того, для начала, загадывая числа, не пользуйтесь каким-либо датчиком случайных чисел, к примеру, одним из наилучших генераторов — монеткой. Интересно убедиться, насколько распределение чисел в ваших представлениях отличается от случайного.

Ну а теперь, после окончания матча (если вы выиграли — поздравляем, если проиграли — ничего страшного,

все равно вы умнее), перейдем к описанию алгоритма, заложенного в нашей программе.

Ходы человека запоминаются последовательно в шести регистрах памяти, которые можно представить в виде одной шестиразрядной ячейки. При каждом новом ходе содержимое разрядов сдвигается влево, и то, что было в последнем разряде, пропадает. В РО записывается ваш текущий ход, который и предстоит отгадать калькулятору. Еще в одном служебном регистре Р4 в зависимости от ситуации (об этом немного позже) устанавливается положение флага *. Если он «опущен», т. е. в Р4 записан нуль, управление передается на блок детерминированного выбора. Он устроен так. Анализируется содержимое первых четырех разрядов нашей ячейки. При этом считается, что если оно равно 0000, 1000, 0101, 0110, 0011, то наиболее вероятен следующий ход «0». Если же в «ячейке» записано 1100, 1001, 0111, 1111, то наиболее вероятным считается ход «1». Понятно, что это предположение носит не очень научный, скорее явно субъективный характер, но ведь и действия человека, играющего в эту игру, тоже субъективны.

Для облегчения анализа приведенные комбинации рассматриваются как числа, записанные в двоичной системе, причем справа налево. Это обстоятельство облегчает использование косвенной адресации, довольно широко применяемой в программе. После перевода наших «обратных» двоичных чисел в привычные десятичные правило для выбора ответа микрокалькулятора можно записать так: если число записанное в «ячейке», равно 0, 1, 6, 10, 12, выбирать нуль, при равенстве числа 3, 5, 9, 14, 15 — единицу. В остальных же ситуациях перейти к блоку случайного выбора.

Кстати, если флаг поднят, то управление сразу же передается на этот блок. Правда, и в этом блоке выбор ответа не совсем случаен. Он зависит от количества единиц и нулей в шести предыдущих ходах. Чем более среди них единиц, тем более вероятно появление единицы и при ответе. То же самое относится и к нулям.

Надо сказать, что использование флага в этой программе определенным образом защищает калькулятор от возможности разгадать его стратегию. В частности, блок

* Флагом в вычислительной технике называют некоторый условный признак, устанавливаемый программно для того, чтобы указать особенности или режим выполнения определенных операций.

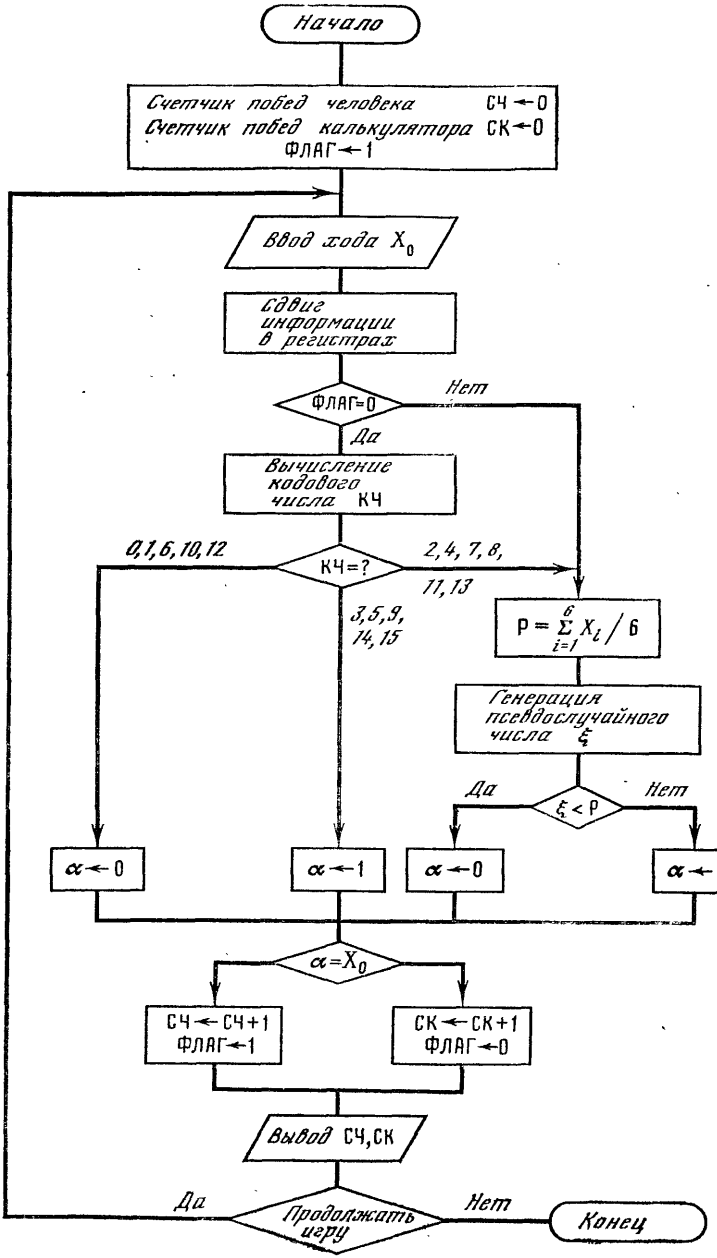

Начало
Счетчик побед человека СЧ←0
Счетчик побед калькулятора СК←0
ФЛАГ←1
Ввод хода X_0
Сдвиг информации в регистрах
ФЛАГ=0
Нет
Да
Вычисление кодового числа КЧ
0,1,6,10,12
КЧ=?
2,4,7,8,
11,13
3,5,9,
14,15
$P = \sum_{i=1}^{6} X_i / 6$
Генерация псевдослучайного числа ξ
Да
ξ < P
Нет
α←0
α←1
α←0
α←1
α=X_0
СЧ←СЧ+1
ФЛАГ←1
СК←СК+1
ФЛАГ←0
Вывод СЧ,СК
Да
Продолжать игру
Нет
Конец

Рис. 47

детерминированного выбора независимо от предыдущих ходов партнера используется только после очередной победы микрокалькулятора. Принципиальная блок-схема алгоритма изображена на рис. 47.

А теперь напомним, что говорилось ранее при описании подобных игр: оптимальной стратегией является чисто случайное чередование ходов. Что же получается? С одной стороны, чисто случайный выбор, а с другой — выигрышная стратегия. Нет ли здесь противоречия? Нет, поскольку наша программа использует именно «неслучайность» выбора человека.

Но ведь, раз ее стратегия хотя бы частично детерминирована, значит, и на нее можно найти управу, иначе говоря, выбрать такую стратегию, при которой наш калькулятор чаще будет в проигрыше. Не будем лишать вас удовольствия провести поиск самостоятельно. Пусть это будет стимулом при игре с машиной.

Впрочем, независимо от исхода поединка можно будет сказать, пользуясь языком спортивных комментаторов: «Проигравших в матче не будет». В любом случае знакомство с игровыми проблемами, изучение методов разрешения игровых конфликтов поможет каждому в главной игре, Игре с большой буквы, которая называется Жизнь.

ОБ АВТОРАХ

Бардадым Виктор Алексеевич — аспирант Московского физико-технического института;

Богданов Дмитрий Данилович — инженер Всесоюзного кардиологического центра;

Бойко Алексей Борисович — инженер Всесоюзного заочного электротехнического института связи;

Данилов Игорь Данилович — кандидат технических наук, старший программист объединения «Росспецгеология»;

Пухначев Юрий Васильевич — кандидат физико-математических наук, доцент Московского физико-технического института;

Романовский Томас Баромеевич — кандидат физико-математических наук, доцент Латвийского государственного университета имени П. Стучки;

Славин Георгий Владимирович — старший методист Госкомитета по профтехобразованию Эстонской ССР.

КИБЕРНЕТИКА

МИКРОКАЛЬКУЛЯТОРЫ В ИГРАХ И ЗАДАЧАХ

Утверждено к печати редколлегией серии
«Кибернетика — неограниченные возможности и возможные ограничения»

Редактор издательства А. А. Боровая. Художник А. М. Драговой
Художественный редактор Н. А. Фильчагина
Технические редакторы Т. С. Жарикова, Т. А. Калинина
Корректоры Н. Г. Васильева, Е. В. Шевченко

ИБ № 31660

Сдано в набор 13.05.86. Подписано к печати 21.07.86. Т-15584. Формат 84×108¹/₃₂. Бумага типографская № 2. Гарнитура обыкновенная новая. Печать высокая Усл. печ. л. 8,4. Усл. кр. отт. 8,61. Уч.-изд. л. 8,8. Тираж 182 000 экз. (2-й Завод 100001—182000 экз.). Тип. зак. 2954. Цена 55 коп.

Ордена Трудового Красного Знамени издательство «Наука»
117864 ГСП-7, Москва В-485 Профсоюзная ул., 90

2-я типография издательства «Наука» 121099, Москва, Г-99, Шубинский пер., 6