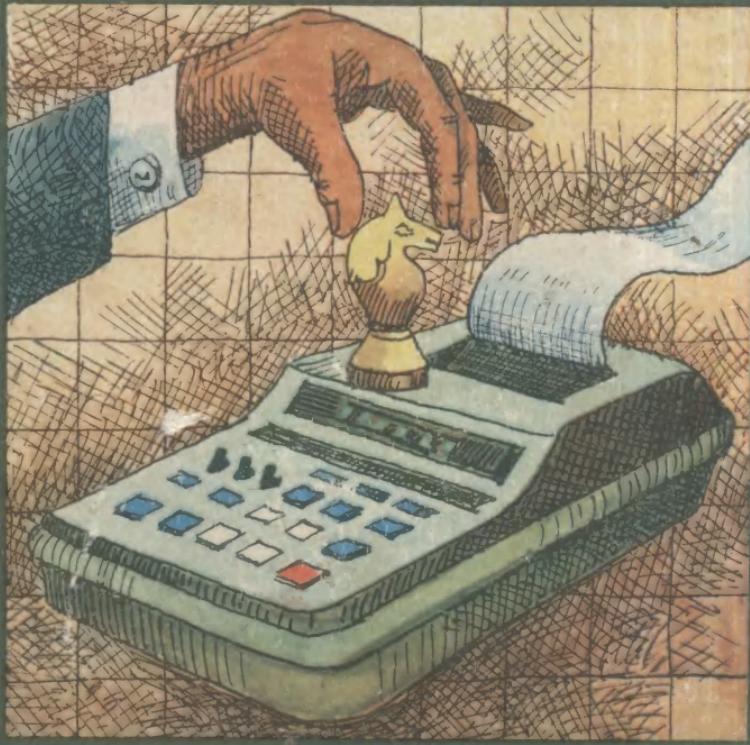


Я. К. Трохименко, Ф. Д. Любич

МИКРОКАЛЬКУЛЯТОР,
ВАШ ХОД!



Я.К.Трохименко, Ф.Д.Любич

МИКРОКАЛЬКУЛЯТОР,
ВАШ ХОД!



Москва «Радио и связь» 1985

ББК 32.97

Т 76

УДК 681.321

Трохименко Я. К., Любич Ф. Д.

Т 76 Микрокалькулятор, Ваш ход! — М.: Радио и связь, 1985. — 224 с., ил.

40 к. 100000 экз.

В популярной форме рассмотрены особенности и методика составления программ работы микрокалькуляторов. Показано их применение для решения разнообразных задач, встречающихся в повседневной жизни. Рассмотрены, в частности, решения типовых математических задач, составление оптимального рациона питания и расчет биоритмов, моделирование игровых автоматов и движения автомобиля, полноправное участие микрокалькуляторов в разнообразных играх.

Для широкого круга читателей, интересующихся возможностями применения микрокалькуляторов.

Т $\frac{2405000000 - 121}{046(01) - 85}$ 124 – 85

ББК 32.97

6Ф0.1

Р е ц е н з е н т ы: доктор техн. наук проф. Е. П. Балашов,
доктор техн. наук проф. С. Д. Пашкеев,
кандидат техн. наук В. А. Балыбердин

Редакция литературы по электронной технике

©Издательство "Радио и связь", 1985

ОГЛАВЛЕНИЕ

Предисловие	4
Глава 1. ЭВМ в кармане	8
1.1. Картофель и алгоритм	8
1.2. Лингвистика карманных ЭВМ	19
1.3. Как работает микрокалькулятор	26
1.4. Искусство программирования	37
1.5. Будьте внимательны!	46
Глава 2. С микрокалькулятором сквозь "дебри" математики	56
2.1. Как обойтись без таблиц?	56
2.2. В поисках корня	65
2.3. Сколько было винограда, или как решить систему уравнений?	78
2.4. Немного статистики	89
2.5. Блуждание в тумане, или оптимизация параметров	101
Глава 3. Наука и жизнь.	114
3.1. Подарок молодым хозяйствам	114
3.2. Ваш день рождения и биоритмы	126
3.3. Покупать ли шкаф?	134
3.4. "Регата" на клавишах	145
3.5. Берегите автомобиль!	158
Глава 4. Микрокалькулятор, Ваш ход!	170
4.1. Может ли мыслить микрокалькулятор?	170
4.2. Игры с камешками	179
4.3. Микрокалькулятор играет в шахматы	193
4.4. "Коктейль" из стратегий	207
4.5. Игры со случаем	214
Список литературы	223

Хвала человеку!
Возьми, оглянись —
Сколько прошел он,
мечтая!
За механизмом
творит
механизм,
Себя самого удивляя.
Пожнет он
и снова растит семена,
А машину за то и ценят,
Что, его без конца заменяя,
она
До конца никогда не заменит.

Вл. Котов

ПРЕДИСЛОВИЕ

Совсем недавно электронные вычислительные машины использовались сравнительно небольшим кругом специалистов. Сегодня положение принципиально изменилось, так как каждому доступны простейшие малогабаритные ЭВМ, называемые микрокалькуляторами. Микрокалькуляторы, предназначенные для школьников, выполняют лишь четыре арифметических действия над числами, а наиболее совершенные из современных программируемых микрокалькуляторов карманных габаритов по своим возможностям превосходят многие универсальные ЭВМ первых поколений, занимавшие несколько просторных помещений. Можно без особого риска предсказать, что уже в ближайшем будущем внедрение микрокалькуляторов (осо-

менно программируемых) и персональных микро-ЭВМ не только повлияет на методику преподавания в средней и высшей школах, но также изменит наш подход к решению многих практических задач.

В этой книге, предназначеннной для широкого круга читателей, не рассматривается внутреннее устройство микрокалькуляторов, представляющее интерес только для специалистов. Задача книги – помочь читателю, не имеющему достаточной подготовки по вычислительной технике и программированию ЭВМ, в эффективном использовании микрокалькуляторов (в основном – программируемых) для решения разнообразных практических задач. Особое внимание в книге уделено участию простейших программируемых микрокалькуляторов в качестве полноправных и "удачливых" партнеров в различного рода играх – область, которую совсем недавно относили к привилегиям человека или, по крайней мере, универсальных ЭВМ с колоссальной памятью и быстродействием.

В книге сравнительно небольшого объема трудно подробно рассмотреть все стороны возможных применений микрокалькуляторов, но авторы могут считать свою задачу выполненной, если книга побудит читателя к развитию приведенных примеров и самостоятельному поиску.



1

ГЛАВА

20 мин

ЭВМ В КАРМАНЕ

**КАРТОФЕЛЬ И АЛГОРИТМ
ЛИНГВИСТИКА КАРМАННЫХ ЭВМ
КАК РАБОТАЕТ МИКРОКАЛЬКУЛЯТОР
ИСКУССТВО ПРОГРАММИРОВАНИЯ
БУДЬТЕ ВНИМАТЕЛЬНЫ!**

1.1. Картофель и алгоритм

Микрокалькуляторы предназначены для вычислений или, другими словами, выполнения определенных операций над числами. Однако сами по себе вычисления, за исключением учебных упражнений, не являются самоцелью и служат лишь средством решения практических задач. Одна и та же задача может быть решена, как правило, различными способами, а требуемое количество и сложность вычислительных или других операций определяются выбранным способом. В простейших случаях план решения конкретной задачи составляют мысленно, но чаще приходится прибегать к его изложению на бумаге с использованием математических моделей.

Математику (греческое слово "матема" означает наука) не случайно называют языком природы – ее предметом, по определению Ф. Энгельса, являются количественные отношения и пространственные формы окружающего мира. Физические (реально существующие) явления и причинно-следственные связи между ними приближенно моделируют различными способами с помощью словесных описаний, физических макетов и математических моделей (множеств геометрических символов и условных знаков, связанных между собой по определенным правилам и отображающих числа и отношения между ними).

Требуемая точность отображения свойств моделируемых объектов зависит от назначения модели. Изображение автомобиля (рис. 1,*a*) образовано геометрическими символами в виде линий и, строго говоря, не является математической моделью. Бессспорно, такая модель более приближенно отображает конструкцию автомобиля, чем инженерный чертеж, но ее точность соответствует назна-

чению: показать общий вид автомобиля. А вот "генеалогическое древо", представленное на рис.1, б, точно отображает отношение типа "кто чей ребенок" в семействе и, несмотря на некоторую вольность формы, относится к математическим моделям, называемым графами и описывающим отношения между множеством объектов.

Чаще используют алгебраические (буквенные) модели, составленные по определенным правилам из букв и условных знаков, отображающих числа и операции над ними. Примером может служить математическая модель в виде формулы $2 \times 2 = ?$ (рис. 1, в), в которой знаком вопроса обозначаем неизвестный результат умножения. Такая формула в общем виде моделирует один из способов подсчета числа объектов в двух множествах, содержащих по два объекта. Могущество математики и заключается в общности (абстрактности) математических моделей, позволяющих отобразить бесконечное многообразие отношений между явлениями природы с помощью конечного числа символов.

Составление плана решения большинства практических задач в основном сводится к поиску таких математических моделей, условий и путей решения задачи, которые позволяют свести исходную проблему к решению типовых мате-

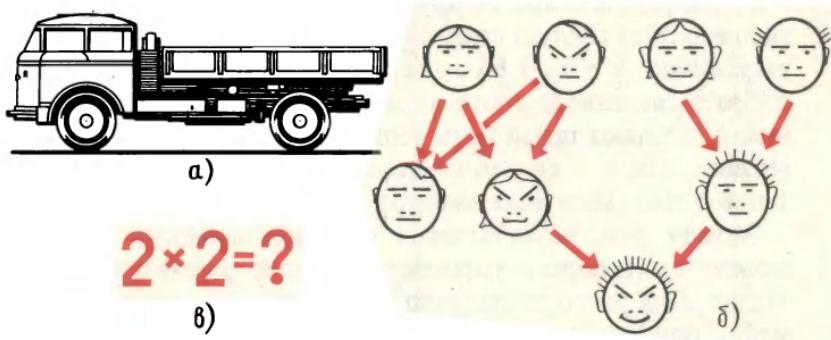


Рис. 1. Математические модели

матических задач. Процесс решения задачи в общем случае состоит из ряда взаимосвязанных этапов: 1) формулировка задачи, заключающаяся в выборе математической модели, связывающей исходные данные с требуемым результатом решения; 2) выбор метода решения, заключающегося в таких преобразованиях исходной модели, которые позволяют получить искомый результат решения с допустимой точностью; 3) выбор способа (алгоритма) решения задачи, обеспечивающего получение искомого результата с минимальными затратами времени или материальных средств; 4) выполнение предусмотренных выбранным алгоритмом операций для получения искомого результата; 5) проверка соответствия полученного результата требованиям задачи (проверка точности решения).

Возможность упрощения решения задачи при ее удачной формулировке наглядно видна на примере известной задачи о птице, которая должна перелететь с одного столба на другой по кратчайшему пути, коснувшись земли. Непосредственная математическая формулировка этой задачи сводится к заданию в качестве исходных данных высот a и b столбов и расстояния между ними d при искомом расстоянии x от одного из столбов до точки касания земли (рис. 2, а), соответствующему кратчайшему расстоянию $h = h_1 + h_2$. При такой формулировке для определения кратчайшего пути следует согласно теореме Пифагора составить выражение $h = h_1 + h_2 = \sqrt{a^2 + x^2} + \sqrt{b^2 + (d - x)^2}$ и подобрать значение x , соответствующее минимальному значению h . Однако такой путь решения задачи достаточно громоздок, так как связан с многократным вычислением значений h для различных значений x .

Между тем, незначительно изменив исходную модель, можно существенно упростить решение рассматриваемой задачи. Для этого достаточно начертить зеркальное отображение одного из столбов (рис. 2, б). В этом случае длина кратчайшего пути $h = \sqrt{(a+b)^2 + d^2}$, а из подобия тре-

угольников на чертеже искомое значение $x = ad/(a + b)$. Таким образом, изменение формулировки задачи определило и выбор более удачного метода ее решения.

Методом называют общий подход к решению определенного класса задач, причем различают прямые (точные) и косвенные (приближенные) методы. Прямыми называют методы, обеспечивающие получение результата после выполнения заранее известного числа определенных операций. В косвенных методах число операций заранее неизвестно, а вычисления прекращают после получения результата с допустимой точностью.

Алгоритмом называют описание способа решения конкретной задачи в виде конечной последовательности однозначных описаний выполнимых операций, приводящих к искомому результату. В этом определении слово "выполнимый" означает, что исполнитель алгоритма может выполнять операцию имеющимися в его распоряжении средствами. В случае, когда описание операции невыполнимо (в частности, исполнитель не знает, как это сделать), то его заменяют последовательностью описания более простых и выполнимых операций. Поэтому в зависимости от знаний

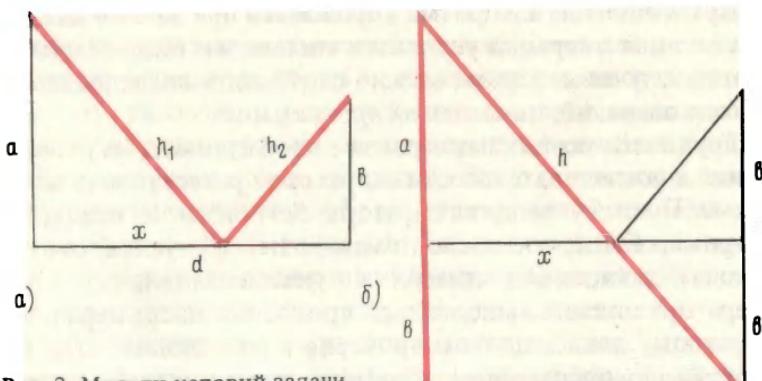


Рис. 2. Модели условий задачи о вороне

и возможностей исполнителя решение задачи выбранным методом отображают алгоритмами, содержащими различные последовательности описаний операций.

Алгоритм представляют словесно-формульным описанием, схемой или программой. *Словесно-формульное описание* обычно составляют из обозначенных порядковыми номерами описаний отдельных операций (шагов или блоков алгоритма) с указанием последовательности их выполнения. Такие указания необходимы при переходах к выполнению операции, не следующей по порядку. Если переход зависит от соответствия результата выполнения предыдущей операции некоторому условию, то его называют *условным*, в противном случае – *безусловным*.

Более наглядно представление алгоритма в виде *схемы*: описания операций проверки заданных условий заключают в ромбы с выходом "Да" или 1 при выполнении и "Нет" или 0 при невыполнении проверяемого условия; описания остальных операций заключают в прямоугольники, а прекращение решения задачи обозначают овалом. Все такие блоки схемы соединяют между собой линиями со стрелками, указывающими последовательность выполнения операций.

Представление алгоритма упрощается при замене каждого описания операций *условным символом (оператором)*. В этом случае алгоритм можно отобразить последовательностью операций, называемой *программой*.

Трудности возникают в связи с необходимостью отображения в последовательности операторов разветвлений алгоритма. Поэтому вводят операторы *безусловного перехода*, например БП *A*, где после команды БП (безусловного перехода) записывают символ *A*, указывающий, с какого оператора должна выполняться программа после перехода. Переходы, зависящие от проверяемого условия, обычно отображают операторами *условного перехода*, занимающими также два шага программы и состоящими из символа

проверяемого условия (например, $x = 0$), после которого записывают указатель A перехода при невыполнении (иногда при выполнении) проверяемого условия. Если это условие удовлетворяется, то программа продолжает выполняться с оператора, записанного после оператора условного перехода.

В программах используют один из двух основных способов передачи управления при переходах. В первом из них указателем A перехода (передачи управления) является адрес (символ порядкового номера) оператора программы, которому передается управление. Во втором указателем A перехода служит определенный символ (метка), записываемый также перед тем оператором, которому передается управление при переходе.

Кстати, эти указатели перехода применяют не только в программах. В книгах можно встретить указатели перехода в виде адресов (например, "см. стр. 106" или см. рис. 22"), так и в виде меток-звездочек или цифр для перехода к примечаниям.

После решения задачи в соответствии с выбранным алгоритмом иногда оказывается необходимым проверять точность решения. Это относится к случаям, когда при выборе формулировки математической модели, метода и алгоритма нельзя полностью учесть особенности решения задачи – например, различного рода погрешности, зависящие от выбранного метода решения, и операционные ошибки, возникающие при выполнении операций. Часто проверка результата точности оказывается более существенной, чем сам результат, так как позволяет оценить преимущества и недостатки выбранного пути решения задачи. Следует добавить, что хорошо составленный алгоритм позволяет не задумываясь ("механически") решать задачу и, следовательно, может служить основой для полной или частичной автоматизации решения задачи при помощи автоматов.

Попробуем показать это на следующей элементарной ситуации, которая иногда встречается в быту. Вам, например, потребовалось сварить картофель. В общем случае алгоритм решения задачи описывается словами "сварить картофель". Если вы впервые столкнулись с этой задачей, то это описание будет непонятным и, следовательно, невыполнимым. Для получения искомого результата (вареного картофеля) можно воспользоваться двумя методами: советами специалиста или поваренной книгой.

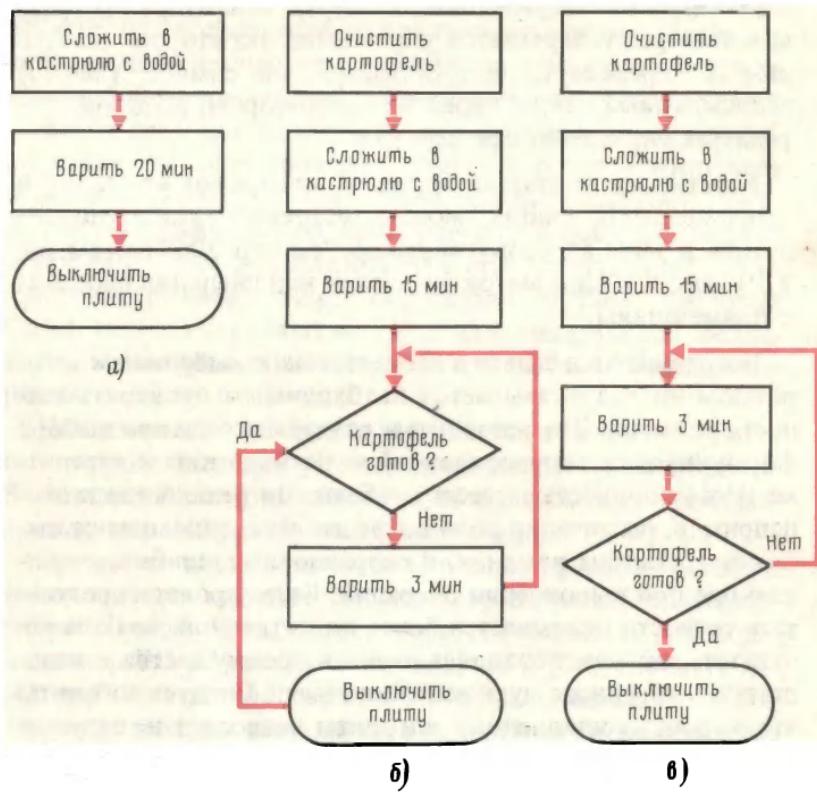


Рис. 3. Схемы алгоритмов решения задачи о варке картофеля

Узнав, что картофель варится около 20 мин, можно составить для решения задачи следующий алгоритм:

1. Сложить картофель в кастрюлю с водой.
2. Варить 20 мин.
3. Выключить плиту.

Несмотря на простоту схемы этого алгоритма (рис. 3,*a*), попытка его реализации может окончиться неудачей, когда картофель окажется сырьеватым или разварится. В сборнике алгоритмов решения подобных задач, называемом поваренной книгой, есть следующее описание: "очистить картофель, сложить в кастрюлю с водой и варить до готовности". В этом описании непонятны слова "варить до готовности", но в быту готовность картофеля проверяется очень просто; если картофель легко протыкается, то он готов. Используя эту информацию, можно составить следующий алгоритм:

1. Очистить картофель.
2. Сложить в кастрюлю с водой.
3. Варить 15 мин.
4. Если картофель готов, то выключить плиту, иначе перейти к п. 5;
5. Варить 5 мин.
6. Если картофель готов, то выключить плиту, иначе перейти к п. 7;
7. Варить 5 мин.
8. Если картофель готов, то выключить плиту, иначе перейти к п. 9;
9. Варить 5 мин.
10. Выключить плиту.

Как видно, в нем учтены все последовательные операции варки картофеля.

Представим этот алгоритм программой, выбрав в качестве операторов начальные буквы описания операций Оч, Сл и В N , где N — число минут варки картофеля. Прекращение решения задачи обозначим оператором С/П ("стоп/пуск"), рассудив, что для повторного решения задачи сле-

дует прекратить ее предыдущее решение. При таком выборе операторов первый из алгоритмов (рис. 3, а) отображался весьма короткой программой (Сл В20 С/П), но в силу доказанных недостатков этого алгоритма такая программа имела чисто теоретическое значение.

Для представления программой второго алгоритма следовало выбрать символ оператора с проверкой условия "Картофель готов?" Остановимся на символе $x = 0 A$, где буква x определяет отклонение состояния картофеля от готового, а буквой A обозначен порядковый номер (адрес) оператора, с которого должно продолжаться выполнение программы в случае неготовности картофеля. Так как при готовности должен выполняться следующий оператор программы, то определенный таким образом оператор описывается условным предложением: "если картофель готов, то перейти к следующей по порядку операции, иначе перейти к операции с адресом A . С учетом сокращенного обозначения такого оператора последний из составленных алгоритмов отображался программой

Оч Сл В15 $x=0$ 7 С/П В5 $x=0$ 11 С/П В5 $x=0$ 15 В5 С/П.

Заметим, что длину этой программы можно сократить при использовании оператора $x \neq 0 A$ ("если картофель не готов, то перейти к следующей операции, иначе перейти к операции с адресом A "). Но такая замена не изменяет существа алгоритма и нужно искать лучшее решение. Попробуем вынести один из одинаковых фрагментов отдельно в виде *подпрограммы*, а в исходной программе на место каждого из таких фрагментов записать оператор ПП A обращения к подпрограмме с адресом A начального оператора подпрограммы. Окончив подпрограмму оператором В/О, означавшим, что после выполнения подпрограммы следует возвратиться обратно к тому месту программы, откуда выполнено обращение к подпрограмме, получим следующую программу:

Оч Сл В15 ПП 12 ПП 12 ПП 12 С/П $x = 0$ 10 В5 В/О

К сожалению, введение подпрограммы не привело к сокращению длины программы, но можно заметить другую возможность — сохранить лишь один из повторяющихся фрагментов, но выполнять его требуемое число раз, организовав цикл. Для подсчета числа выполнений такого фрагмента используем счетчик, обозначаемый символом $k = k + 1$ и означающий, что после каждого выполнения такого счетчика его содержимое увеличивается на единицу. При начальном значении $k = 0$ после n выполнений такого счетчика его содержимое становится равным $k = n$. Для прекращения варки картофеля после трех выполнений такого оператора-счетчика решаем ввести оператор $x = 3 \text{ } 4$, передающий оператору с адресом 4 в начале цикла при $k < 3$ или оператору С/П при $k = 3$:

Оч Сл В10 В5 $x = 0 \text{ } 11$ В5 $k = k + 1$ $x = 3 \text{ } 4$ С/П.

Согласно составленной программе варка картофеля прекращалась оператором $x = 0 \text{ } 11$, если готовность достиглась за 15, 20 или 25 мин. Если картофель не был готов за это время, то он варился еще 5 мин, после чего плита выключалась.

Размышления над этой программой приводят к более плодотворной идеи — прекращать варку картофеля только после его готовности. Для большей точности результата варки заменим оператор В5 оператором В3 (варить 3 мин) и введем после него оператор безусловного перехода БП 4 к началу цикла:

Оч Сл В15 $x = 0 \text{ } 9$ В3 БП 4 С/П.

Этой программе соответствовала схема алгоритма, показанная на рис. 3, б, и следующее описание:

1. Очистить картофель.
2. Сложить в кастрюлю с водой.
3. Варить 15 мин.
4. Если картофель готов, то перейти к п. 6, иначе к п. 5.
5. Варить 3 мин и перейти к п. 4.

6. Выключить плиту.

Этот алгоритм удачнее предыдущих, так как процесс варки прекращается только после готовности картофеля. Стремление к изъятию лишнего в концовке алгоритма, которая содержала несколько переходов, привело к перестановке в программе оператора ВЗ в начало цикла. Окончательный ее вариант получится

Оч Сл В15 В3 $x = 0 \ 4$ С/П,

соответствовавший показанной на рис. 3, в схеме и следующему описанию:

1. Очистить картофель.
2. Сложить в кастрюлю с водой.
3. Варить 15 мин.
4. Варить 3 мин.
5. Если картофель готов, то перейти к п. 6, иначе к п. 4.
6. Выключить плиту.

Следует подчеркнуть, что в поисках решения задачи о картофеле использованы различные методы. Первый (рис. 3, а) и второй его алгоритмы реализуют прямой метод решения, тогда как два последних алгоритма (рис. 3, б и в) — косвенный метод последовательных приближений, отображаемый в алгоритмах замкнутыми циклами, охватывающими повторно выполняемую последовательность операций (*итерацию*). Однако очевидные преимущества косвенных методов еще не означают их преимущество в общем случае. Убедительный пример — задача о птице, где применение прямого метода упрощает решение.

Выполнение операций, записанных в последнем алгоритме, можно поручить уже автоматам. Это приводит к мысли о том, что составление программ, представляющих алгоритмы, подобно составлению текста на обычных языках. В таком алгоритмическом языке, "понятном" автомату, функции слов выполняют команды-операторы, а правила, определяющие допустимые сочетания операторов, соответству-

ют синтаксическим правилам обычного языка. Более того, различаются два языковых уровня для автоматов. Программы, составленные из вводимых нажатием кнопок сложных команд-операторов (например, "Сварить картофель", "Взять 3 яйца", "Приготовить кофе с молоком" и т. п.), соответствуют входному (внешнему) языку автомата. Однако каждая из таких команд выполнялась бы автоматом в виде последовательности более простых операций, соответствовавших микропрограмме реализации команды на внутреннем (машинном) языке автомата.

В заключение отметим, что обращение к алгоритмам и программам на "кухонные" темы не является необычным. Автор известной многотомной монографии по программированию ЭВМ [6] признался на страницах своей книги, что он очень хотел назвать ее "Поваренной книгой для программистов".

1.2. Лингвистика карманных ЭВМ

Цифровые ЭВМ, к которым относятся и микрокалькуляторы, выполняют операции над такими абстрактными объектами, как числа (называемые в этом случае *операндами*), и, казалось бы, не имеют ничего общего с кухонными автоматами, выполняющими операции над вполне материальными субстанциями в виде продуктов. Однако то обстоятельство, что кухонные автоматы выполняют операции над отдельными (дискретными) порциями продуктов, и объединило их с ЭВМ в едином семействе дискретных автоматов, программирование которых подчиняется общим закономерностям.

В зависимости от сложности автоматически выполняемых операций микрокалькуляторы любого типа можно отнести к одному из трех следующих основных классов.

Арифметические микрокалькуляторы по командам, подаваемым нажатием клавиш, автоматически выполняют арифметические операции (сложение, вычитание, умножение, деление) и иногда вычисляют простейшие функции (обратную величину, квадратные корни, проценты).

Специализированные (в основном инженерные и коммерческие) микрокалькуляторы по командам, подаваемым нажатием клавиш, кроме выполнения арифметических операций, вычисляют ряд элементарных и специальных функций.

Программируемые микрокалькуляторы, кроме выполнения операций над числами при нажатии клавиш (*обычный режим*), автоматически выполняют вычисления по программе, заранее введенной в микрокалькулятор в режиме *программирования*.

На передних панелях представителей каждого из этих классов (рис. 4) расположены индикатор для десятичных представлений чисел и пульт управления с клавишами и переключателями для ввода команд (операторов). Числа представляются на индикаторах в *естественной форме* с запятой, фиксированной между целой и дробной частями, или в *показательной форме* с плавающей запятой $a = M \cdot 10^n$, где M – мантисса, а n – порядок числа. Для микрокаль-

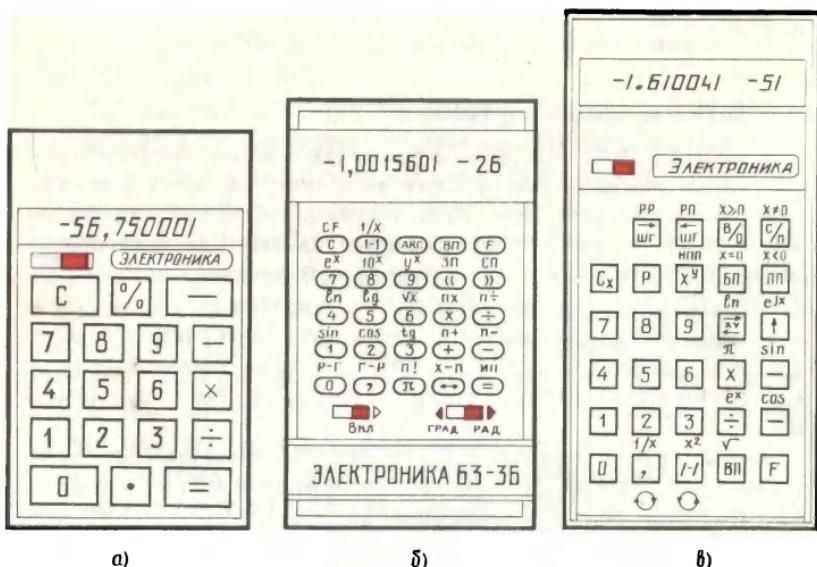


Рис. 4. Микрокалькуляторы типа:
а – "Электроника Б3-23"; б – "Электроника Б3-36"; в – "Электроника Б3-21"

куляторов характерна стандартная показательная форма представления результатов вычислений, при которой целая часть мантиссы содержит одну не равную нулю цифру ($1 \leq M < 10$).

На клавиатуре пульта управления обозначены символы команд, образующие алфавит входного языка микрокалькулятора. Из одного или допустимых сочетаний нескольких элементов алфавита формируются слова входного языка в виде служебных команд и операторов программы. Допустимые сочетания операторов определяются синтаксическими правилами входного языка. Операторы, из которых составляется программа, относятся к одной из нескольких основных групп операторов.

Группа операторов набора чисел во всех микрокалькуляторах содержит операторы набора цифр от 0 до 9, десятичного разделительного знака (запятой или точки) и изменения знака числа, обозначаемого символами $/-$, $+/-$ или CHS. В микрокалькуляторах, допускающих показательную форму представления чисел, к этой группе относится оператор ввода порядка (обозначаемый символом ВН, n, ЕЕ или ЕЕХ), оператор набора числа $\pi = 3,1415927$ и некоторые другие.

Группа функциональных операторов обязательно содержит двухместные (выполняемые над двумя операндами) арифметические операторы $+$, $-$, \times и \div . К двухместным относится также оператор X^Y или Y^X для вычисления степенной функции общего вида. Одноместными называют операторы вычисления функций одного аргумента — простейших степенных функций $y = x^{-1} = 1/x$, $y = x^2$, $y = -x^{1/2} = \sqrt{-x}$, логарифмических, показательных, тригонометрических, обратных тригонометрических и других элементарных функций. К одноместным относится также оператор вычисления факто-риала $x! = x(x - 1) \cdot (x - 2) \dots 2 \cdot 1$ (см. рис. 4, б) и оператор e^{jx} (см. рис. 4, в) для одновременного вычисления $\sin x$ и $\cos x$.

Словари некоторых микрокалькуляторов содержат и многоместные функциональные операторы, например оператор вычисления вещественных корней квадратного уравнения во входном языке микрокалькулятора "Электроника Б3-32".

Большинство микрокалькуляторов содержит запоминающее устройство (память) для хранения чисел, а их словари содержат операторы обращения к памяти. К ним относятся операторы занесения числа в память, обозначаемые символами ЗАП, П, $x \rightarrow m$ или STO,

и вызова числа из памяти, обозначаемые, например, символами ИП, СЧ, $m \rightarrow x$ или RCL . Если в памяти может одновременно находиться несколько чисел, то оператор обращения к памяти содержит также номер регистра (устройства для хранения одного числа), в который заносится или из которого вызывается число. Словари некоторых микрокалькуляторов содержат операторы для выполнения двухместных операций над содержимым памяти и высвечиваемым на индикаторе числом, а также операторы для стирания содержимого памяти.

Для уменьшения количества клавиш на пульте управления часть команд обозначают над или под клавишами и для ввода таких команд предварительно нажимают префиксную клавишу, часто обозначаемую символом F . К префиксным относятся, например, команда arc в некоторых входных языках, ввод которой перед вводом оператора вычисления тригонометрической функции приводит к вычислению соответствующей обратной тригонометрической функции.

Группа "синтаксических" операторов связана с синтаксическими особенностями входного языка, от выбора которых в значительной мере зависит сложность изготовления и, следовательно, стоимость микрокалькулятора.

Обычные правила записи математических формул для вычислений (их называют арифметическими выражениями независимо от выполняемых операций) допускают дробные черты, радикалы и другие символы с протяженными элементами, облегчающими образное восприятие формул человеком, например

$$y = a - b / \sqrt{\operatorname{tg}(\alpha - \beta)} .$$

Для вычислений по таким формулам с помощью дискретных автоматов символы с протяженными элементами должны быть заменены операторами, занимающими один шаг программы, например $a - b \div \sqrt{\operatorname{tg}(\alpha - \beta)} = ,$

где буквами обозначены набираемые или вызываемые из памяти операнды, а оператор-вывода результата записывается в конце программы.

Основной недостаток правил подобной формы арифметических выражений, называемой *естественной* алгебраической записью при их использовании для синтаксиса входного языка микрокалькулятора, связан с необходимостью запоминания операторов, которые могут быть выполнены только после ввода следующих за ними операнда и оператора, ввод которого сигнализирует об окончании ввода операнда и возможности выполнения предыдущего оператора. Поэтому во входных языках микрокалькуляторов обычно используют синтаксическое правило записи одноместных операторов после операндов, например

$$a - b \div (\alpha - \beta) \operatorname{tg} \sqrt{} = .$$

При использовании подобной формы, называемой *простой* алгебраической записью, одноместные операторы выполняются без их запоминания непосредственно после ввода, что упрощает конструкцию микрокалькулятора.

Входные языки некоторых микрокалькуляторов (например, типа "Электроника С3-15") автоматически учитывают необходимость выполнения (старшинство) операций умножения и деления перед сложением и вычитанием. Однако чаще для учета старшинства операций используют только операторы скобок, например

$$a - (b \div (\alpha - \beta) \operatorname{tg} \sqrt{}) = .$$

Такую форму называют простой *скобочной* записью. Для упрощения конструкции входные языки многих микрокалькуляторов с алгебраическим синтаксисом предусматривают простую *бесскобочную* запись, при которой старшинство операций не учитывается. При вычислениях на таких микрокалькуляторах для учета старшинства операций приходится обращаться к памяти или, когда это возможно, предварительно, преобразовывать арифметическое выражение, например

$$\alpha - \beta \operatorname{tg} \sqrt{} \div b = 1/x / - / + a = .$$

Преимущества простой записи одноместных операторов после операндов привели к использованию во входных языках некоторых микрокалькуляторов синтаксического правила *обратной* (называемой также *обратной* или *инверской* польской записью) записи

ратор, а символом \dagger обозначен оператор, разделяющий операнды.

В обратной записи рассматриваемое арифметическое выражение принимает вид

$$a \dagger b \dagger \alpha \dagger \beta - \operatorname{tg} \sqrt{\div} \div - .$$

В этом случае конструкция микрокалькулятора должна обеспечивать запоминание четырех операндов в процессе вычислений. Если допустимо запоминание не более трех операндов, то рассматриваемое арифметическое выражение записывается как

$$b \dagger \alpha \dagger \beta - \operatorname{tg} \sqrt{\div} \div / - / \dagger a + .$$

Если же возможно запоминание не более двух операндов, то арифметическое выражение следует записать

$$\alpha \dagger \beta - \operatorname{tg} \sqrt{\dagger} b \div 1/x \div / - / \dagger a + .$$

Однако в любом случае при обратном синтаксисе входного языка все операторы выполняются непосредственно после их ввода, что исключает необходимость их запоминания и упрощает конструкцию микрокалькулятора. Преимущества обратного синтаксиса входных языков микрокалькуляторов не случайны. По назначению эти языки соответствуют командным формам обычных языков, которые также отличаются обратным порядком слов. Так, естественная запись \sqrt{a} соответствует в русском языке повелительной форме "Извлеките корень из числа a !", тогда как обратная запись $a \sqrt{\dagger}$ соответствует более категоричной команде "Из числа a корень извлечь!". Аналогично алгебраическая запись $a + b =$ описывается словами "Число a сложите с числом b и запишите результат", тогда как обратная запись $a \dagger b +$ соответствует команде "Число a и число b сложить!".

Последняя форма характерна для всех команд, которые должны быть выполнены без промедления. Правила русского языка не запрещают подачу команды типа "Стой, Иванов и Петров!", но момент их выполнения оказывается неопределенным, так как может последовать как команда "Стой, Иванов", так и команда "Стой, Иванов, и Сидоров, и Петров!". Поэтому в командах первым вводят подлежащее, определяющее объект, который должен быть готов к выполнению команды, а затем сказуемое, определяющее действие объекта и служащее сигналом для выполнения команды,

например колонна демонстрантов, стои: или первая шеренга, вперед!".

Таким образом, для словарного запаса входных языков с алгебраическим синтаксисом характерен оператор $=$, функции которого в простейших микрокалькуляторах иногда выполняют операторы $+$ и $-$, а для входных языков со скобочным алгебраическим синтаксисом характерны также операторы скобок $($ и $)$ или, если допустимо хранение более одного дополнительного операнда, $(($ и $)$.

В словарях входных языков с обратным синтаксисом эти операторы отсутствуют, но имеется оператор разделения operandов, обозначаемый символом \dagger или $E\dagger$. Кроме них группа синтаксических операторов содержит операторы стирания (C_x , СК или CLx), высвечиваемого на индикаторе операнда и операторы С или CL стирания всей информации, хранимой в ходе вычислений. К этой группе относится также оператор $X \leftrightarrow Y$ или \leftrightarrow обмена местами первого или второго операнда при двухместных операциях и некоторые другие.

Для программируемых микрокалькуляторов, кроме рассмотренных, характерны операторы управления программой автоматических вычислений. К ним относятся уже известные нам условные операторы вида $x\Pi A$, где A – указатель перехода, а Π – отношение равенства или неравенства между результатом x предыдущей операции и некоторым эталонным числом (обычно $z = 0$). Команду безусловного перехода общего назначения обычно обозначают символом БП или GTO, а команду обращения к подпрограмме – символом ПП, SBR или SRT. Оператор возврата из подпрограммы обозначают символом В/O, RST или RES. Если этот оператор записан в основной части программы, то он передает управление второму ее оператору. При передаче управления с помощью меток словарь программируемого микрокалькулятора содержит символ LbN , где N – номер метки, или метки в виде букв. Адреса в программах указывают либо набором цифр, либо операторами, кодирующими адрес при нажатии соответствующих клавиш.

Программу автоматических вычислений оканчивают оператором С/П, автоматически прекращающим вычисления. Этот же оператор вводят нажатием клавиши при пуске программы автоматических вычислений и при необходимости ее остановки. В зарубежных микрокалькуляторах оператор "стоп/пуск" обозначают символом R/S.

Словари некоторых программируемых микрокалькуляторов содержат оператор Φ или SF установки флага – сигнала о некоторых свойствах operandов или выполняемых операций, являющегося своеобразным "узелком на память". В этом случае установка флага проверяется условным оператором $\Phi?$ или IF FG?, после которого в программу записывают оператор передачи управления (адрес или метку) при установленном или, в некоторых языках, при неустановленном флаге.

Группа служебных команд программируемых микрокалькуляторов содержит команды перехода в режим программирования РП, ПРГ, или LRN, перехода в режим вычисления PP, АВТ или USER, а также операторы, используемые для исправления ошибок при вводе программы автоматических вычислений или для ее редактирования.

1.3. Как работает микрокалькулятор

Микрокалькуляторы отличаются от универсальных ЭВМ в основном лишь скоростью выполнения операций и емкостью запоминающих устройств и имеют весьма сложную конструкцию: простейшие из них содержат несколько тысяч, а более совершенные несколько десятков или даже сотен тысяч транзисторов в интегральном исполнении. Однако пользователю нет необходимости подробно изучать устройство микрокалькулятора, так как для эффективного его использования достаточно изучить связи между нажатиями клавиш и их последствиями. На обобщенной схеме микрокалькулятора (рис. 5) показаны связи между основными функциональными узлами микрокалькулятора: пультом управления, управляющим и запоминающим устройствами, процессором (вычислителем) и индикатором. Отдельным прямоугольником показан источник питания с сухими электрическими элементами или малогабаритным аккумулятором, подзаряжаемым от сети переменного тока через вилку-выпрямитель.

Индикатор содержит ряд знакомест для индикации (высвечивания) десятичных представлений чисел и иногда вспомогательных знаков. Одно или, в микрокалькуляторах с показательной формой представления чисел, два знакоместа предназначены для высвечивания отрицательных знаков и символов, характеризующих режим работы микрокалькулятора, а остальные (от семи до двенадцати) – для высвечивания цифр в десятичных разрядах чисел. Знакоместа

для высвечивания цифр состоят из нескольких (обычно семи) световых элементов и при подаче напряжения на определенную их комбинацию высвечивается требуемая цифра. Запятая высвечивается в цифровых знакоместах или, с помощью дополнительных световых элементов, между ними.

Количество цифровых знакомест определяет количество разрядов и диапазон представления чисел на индикаторе. При естественной форме представления чисел с r цифровыми разрядами этот диапазон ограничен значениями $A_{\min} = 10^{1-r}$ и $A_{\max} = 10^r - 1$. При стандартной показательной форме с r разрядами мантиссы и m порядка диапазон представления чисел значительно шире и ограничен значениями

$$A_{\min} = 1 \cdot 10^{-10^{m+1}} \text{ и } A_{\max} = (10 - 10^{1-r}) 10^{10^m - 1}.$$

При абсолютном значении, меньшем A_{\min} , результат попадает в область машинного нуля и на индикаторе высвечивается нуль. Если результат вычислений по абсолютной величине больше A_{\max} , то он попадает в область машинной бесконечности, возникает переполнение регистров (запоминающих устройств для хранения чисел)

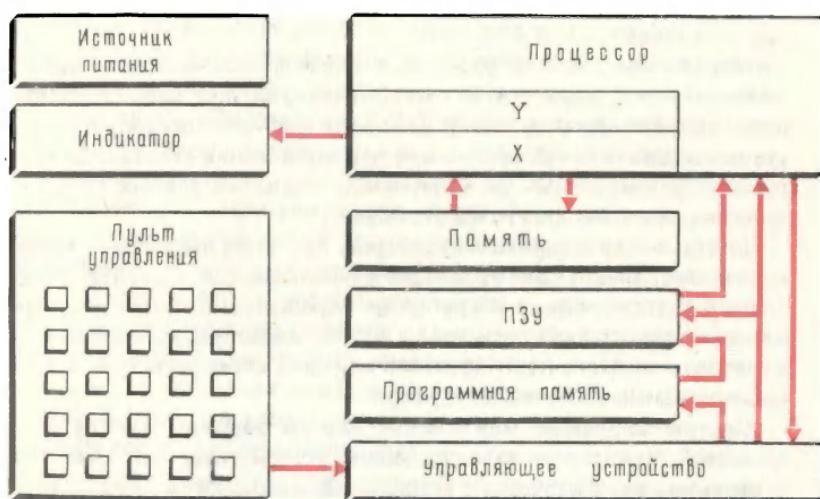


Рис. 5. Обобщенная схема микрокалькулятора

микрокалькулятора и на индикаторе возникает сигнал переполнения – гашение высвечиваемого числа, высвечивание специального символа и т. п.

Управляющее устройство согласует во времени работу всех узлов микрокалькулятора и обычно конструктивно совмещено с процессором. Последний состоит из арифметико-логического устройства, выполняющего арифметические операции над машинными (двоичными или десятично-двоичными) представлениями чисел, и операционного устройства для временного хранения операндов и, при алгебраическом синтаксисе входного языка, кодов двухместных операторов.

Операционное устройство содержит не менее двух (для выполнения двухместных операций) регистров, из которых один обычно обозначают символом Y , а другой называют регистром X , индикаторным или входным. Операнд, хранимый в регистре X , высвечивается на индикаторе с помощью устройства, преобразующего код этого оператора в напряжение, подаваемые на соответствующие световые элементы знакомест индикатора. При числе регистров операционного устройства, большем двух, микрокалькулятор непосредственно вычисляет арифметические выражения со скобками, причем число дополнительных регистров определяет и число "скобок в скобках".

Микрокалькуляторы различных классов отличаются составом запоминающих устройств. Все микрокалькуляторы имеют постоянное запоминающее устройство (ПЗУ) для хранения микропрограмм, управляющих работой процессора при выполнении операторов программы, составленных на внутреннем языке машинных кодов и записанных в ПЗУ при его изготовлении.

Большинство микрокалькуляторов, исключая простейшие арифметические, имеют запоминающие устройства для хранения чисел (память), управляемые операторами обращения к памяти и содержащие от одного-двух регистров в непрограммируемых микрокалькуляторах до нескольких десятков или даже сотен регистров в программируемых микрокалькуляторах.

Программируемые микрокалькуляторы характеризуются *программной памятью* – запоминающим устройством для хранения программы автоматических вычислений, вводимой в режиме программирования. Емкость программной памяти определяется допустимым числом операторов (шагов) программы и составляет

от 50–100 шагов для простейших программируемых микрокалькуляторов до нескольких тысяч для наиболее совершенных. Последние обычно снабжены и внешними (периферийными) устройствами — миниатюрными печатающими приспособлениями, накопителями информации на магнитных карточках и сменными модулями типовых программ. Такие микрокалькуляторы по ряду показателей превосходят многие универсальные ЭВМ первых поколений.

В обычном режиме вычислений нажатиями клавиш при вводе оператора набора десятичного знака управляющее устройство засыпает его код в регистр X операционного устройства, что приводит к высвечиванию этого знака на индикаторе. При вводе оператора обращения к памяти копия содержимого регистра X засыпается в регистр памяти, а при вводе оператора вызова из памяти копия содержимого регистра памяти засыпается в регистр X .

При вводе одноместных и, при обратном синтаксисе входного языка, двухместных операторов управляющее устройство вызывает из ПЗУ и засыпает в процессор микропрограмму выполнения соответствующей операции над операндами, хранящимися в операционном устройстве. Результат операции засыпается в регистр X и высвечивается на индикаторе.

При вводе двухместного оператора в микрокалькулятор с алгебраическим синтаксисом входного языка код этого оператора запоминается в операционном устройстве, а хранившийся там код ранее введенного двухместного оператора вызывает из ПЗУ и засыпает в процессор микропрограмму выполнения соответствующей операции.

Таким образом, синтаксические особенности входного языка микрокалькулятора определяют правила работы и, следовательно, конструкцию его оперативного устройства. Регистры этого устройства соединены между собой (такое соединение называют стеком) и обмениваются своим содержимым в соответствии с определенными правилами. Правила работы операционных стеков удобно отображать эквивалентными схемами, которые могут точно не соответствовать реальной конструкции операционного устройства, но позволяют установить связи между нажатием клавиш и высвечиваемыми на индикаторе представлениями чисел.

Арифметические микрокалькуляторы (см. рис. 4, а) отличаются входными языками с простым бесскобочным алгебраическим синтаксисом, а работа их операционных стеков обычно подчинена следующим правилам:

при наборе операнда до выполнения двухместной операции его код заносится в регистр X без изменения содержимого регистра Y (рис. 6, а);

при наборе операнда после выполнения двухместных операций его код заносится в регистр X , прежнее содержимое которого засыпается в регистр Y (рис. 6, б);

при вводе двухместного оператора в начале программы его код засыпается в регистр Y , куда засыпается и содержимое регистра X (рис. 6, в);

при вводе двухместного оператора после двухместного оператора или после двухместного оператора и операнда его код заносится в регистр Y , выполняется предыдущая операция над содержимым регистров Y и X , ее результат заносится в регистр X , предыдущее содержимое которого засыпается в регистр Y (рис. 6, г). Аналогичные действия выполняются и при вводе оператора $=$, но в этом случае код предыдущего двухместного оператора сохраняется в регистре Y ;

при вводе оператора $=$ после такого же оператора содержимое регистра Y фиксируется и выполняется двухместная операция, код которой хранится в регистре Y , причем operand из регистра Y вводится в операцию вторым (рис. 6, д).

Последнее из приведенных правил обеспечивает, в частности, выполнение нескольких двухместных операций с одним и тем же операндом (константой), например:

$$С \ a \div b = [a/b] \ c = [c/b] \ d = [d/c],$$

где в квадратных скобках указаны высвечиваемые результаты опе-

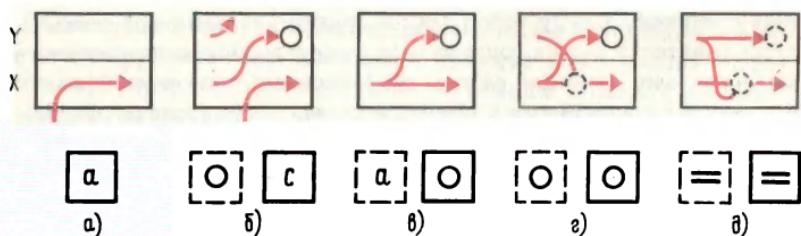


Рис. 6. Схемы работы операционного стека арифметического микрокалькулятора

раций. Это же правило обеспечивает возвведение чисел в целые положительные или отрицательные степени:

$$С \ a \div = [1] = [a^{-1}] = [a^{-2}] = [a^{-3}]$$

или

$$С \ a + b = [a + b] \times = [(a + b)^2] = [(a + b)^3].$$

В микрокалькуляторах ранних выпусков для выполнения подобных вычислений использовался специальный оператор константы К.

Большинство специализированных микрокалькуляторов отличается простым скобочным алгебраическим синтаксисом входного языка. Примером может служить инженерный микрокалькулятор типа "Электроника Б3-36", регистры X и Y которого при наборе (и вводе из памяти) операндов и двухместных операциях работают по рассмотренным правилам (рис. 6, а-д). При вводе одноместного оператора $n!$ вычисления факториала в регистр Y засыпается единица (рис. 7, а), но при вводе остальных одноместных операторов содержимое регистра Y не изменяется (рис. 7, б).

При вводе оператора \leftrightarrow обмена содержимое (операнды) регистров X и Y меняются местами (рис. 7, в). При вводе оператора ((открывающей скобки коды операнда из регистра X и оператора из регистра Y засыпаются также в регистр Z , содержимое которого смещается в регистр T (рис. 7, г). При вводе оператора)) закрывающей скобки над содержимым регистров X и Y выполняется двухместная операция, код которой хранится в регистре Y и стек смещается "вниз" (рис. 7, д).

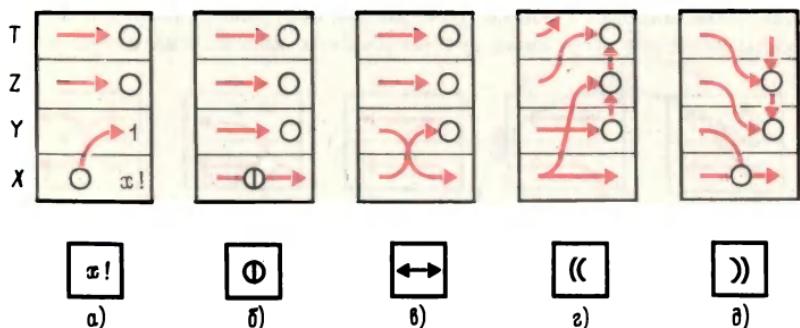


Рис. 7. Схемы работы операционного стека микрокалькулятора "Электроника Б3-36"

Некоторые специализированные и большинство программируемых микрокалькуляторов отличаются обратным синтаксисом входного языка, который несколько непривычен для начинающих пользователей, но позволяет упростить конструкцию микрокалькулятора. Примером может служить микрокалькулятор "Электроника Б3-21" (см. рис. 4, е). Работа его двухрегистрового операционного стека описывается следующими несложными правилами:

при вводе оператора \uparrow и при наборе операнда после выполнения операции содержимое операционного стека смещается "вверх" (рис. 8, а);

при наборе операнда после оператора \uparrow или вызове из памяти код операнда заносится в регистр X без изменения содержимого регистра Y (рис. 8, б);

при вводе двухместных (рис. 8, в) и одноместных (кроме e^{jx}) операторов результат операции замещает прежнее содержимое регистра X без изменения содержимого регистра Y ;

при вводе оператора e^{jx} в регистры X и Y соответственно заносятся $\cos x$ и $\sin x$ (рис. 8, г) и для вычисления $\operatorname{tg} x$ достаточно дополнительно ввести оператор \div ;

при вводе оператора обмена XY содержимое регистров X и Y меняется местами (рис. 8, д).

Аналогичные или близкие правила определяют и работу операционных стеков с большим числом регистров специализированных (например, "Электроника Б3-19М") и программируемых (например, "Электроника Б3-34") микрокалькуляторов с обратным синтаксисом входного языка. При работе программируемых микрокалькуляторов в режиме автоматических вычислений операцион-

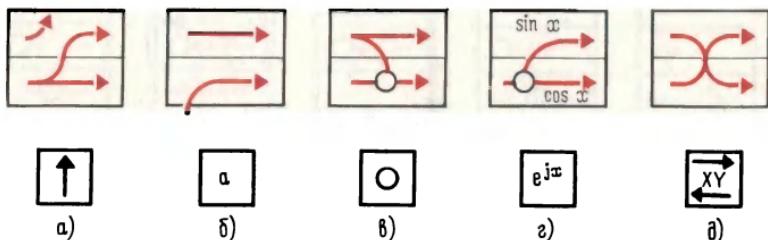


Рис. 8. Схемы работы операционного стека микрокалькулятора "Электроника Б3-21"

ные стеки работают по тем же правилам, что и в обычном режиме, но коды выполняемых операторов считаются из программной памяти.

Работу программируемых микрокалькуляторов рассмотрим более подробно на примере простейшего микрокалькулятора этого класса — первого массового отечественного программируемого микрокалькулятора "Электроника Б3-21".

Память этого микрокалькулятора имеет 13 регистров, из которых семь с номерами от 2 до 8 образуют запоминающее устройство с произвольной выборкой (ЗУПВ). Для засылки содержимого регистра X в N -й регистр ЗУПВ вводят оператор PN (нажимая клавиши с символами P и N), а для вызова содержимого N -го регистра ЗУПВ в регистр X вводят оператор FN . Остальные шесть регистров памяти вместе с регистром X соединены в кольцевой стек (рис. 9). При вводе оператора \circlearrowleft поворота этого стека по часовой стрелке содержимое каждого его регистра засыпается в соседний правый регистр так, что прежнее содержимое регистра X засыпается в регистр $C1$, а содержимое регистра $C6$ засыпается в регистр X . При вводе оператора \circlearrowright содержимое кольцевого стека смещается против часовой стрелки, содержимое регистра $C1$ засыпается в регистр X , а прежнее содержимое последнего — в регистр $C6$. Одноместные операторы $1/x$, x^2 и \sqrt{x} в этом микрокалькуляторе вводят после нажатия префиксной клавиши F , а остальные операторы, обозначенные над клавишами, и операторы поворота стека — после нажатия клавиши P (см. рис. 4, в).

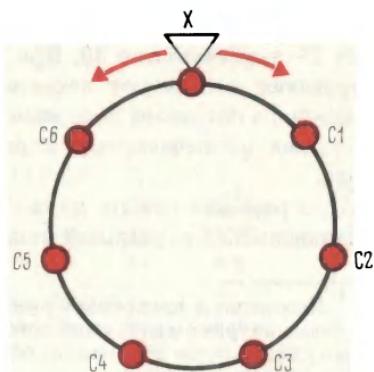


Рис. 9. Схема кольцевого стека памяти

Вычисления в обычном режиме соответствуют правилам работы операционного стека, отраженным на рис. 8, а для вычислений в программируемом режиме вводят служебный оператор РП для перевода микрокалькулятора в режим программирования, после чего нажатиями клавиш вводят в программную память программу автоматических вычислений. Такая программа, кроме оператора С/П в конце основного текста и операторов, используемых в обычном режиме, может содержать команды условных переходов $x = 0$, $x \neq 0$, $x \leq 0$ и $x < 0$, команды безусловного перехода БП, команды ПП обращения к подпрограммам и оператор В/О выхода из них, а также указатели передачи управления (коды адресов), записываемые в программе после операторов условного и безусловного переходов.

Каждому оператору входного языка (кроме служебных команд) присвоены двузначные десятичные коды (табл. 1), частично совпадающие с кодами адресов, вводимых нажатиями тех же клавиш. При вводе программы в режиме программирования на индикаторе в разрядах мантиссы высвечиваются коды трех операторов, введенных последними, а в разрядах порядка высвечивается адрес (счетчик шагов) оператора, введенного последним¹.

Программная память микрокалькулятора емкостью 60 шагов разбита на 10 "страниц" с номерами $a = 0, 1, \dots, 9$, содержащих по шесть операторов с номерами $b = 1, 2, \dots, 6$ на "странице". Сочетание цифр ab и образует адрес оператора, которому передается управление оператором передачи управления, вводимым нажатием определенных клавиш (см. табл. 1). При этом в счетчике шагов адреса с цифрой $b = 6$ высчитываются как $(a + 1)0$ и, например, вместо адреса 26 высвечивается 30. При необходимости в режиме программирования используют оператор НОП для замещения ошибочно введенного оператора программы, а также операторы $\overleftarrow{ШГ}$ и $\overleftarrow{ШГ}$ для смещения высчитываемых кодов программы на шаг вперед или назад.

Программная память по связям между ее ячейками аналогична (независимо от ее реальной конструкции) кольцевому стеку, пово-

¹ Принятая в настоящей книге система адресации несколько отличается от рекомендуемой инструкцией по эксплуатации микрокалькулятора, где для кодов операторов $b = 1, 2, \dots, 6$, а для адресов $b = 0, 1, \dots, 5$.

Таблица 1

Адреса и коды операторов входного языка
микрокалькулятора "Электроника Б3-21"

Адрес	Код	Клавиши	Оператор	Адрес	Код	Клавиши	Оператор
01		P 0		52	52	F 5	F 5
02		F 0		53	53	P /-	
03	03	P \uparrow	e^{jx}	54	54	5	5
04	04	0	0	55	55	F /-	x^2
05		F \uparrow		56 (60)	56	/-	/-
06	06	\uparrow	\uparrow		58	БП	БП
(10)							
11	11	P 1	P 1		59	P БП	$x = 0$
12	12	F 1	F 1	61	61	P 6	P 6
13	13	P XY	ln	62	62	F 6	F 6
14	14	l ^p	1	63		P ВП	
15		F XY		64	64	6	6
16	16	XY	XY	65	65	F ВП	✓
(20)							
21	21	P 2	P 2	66 (70)	66	ВП	ВП
22	22	F 2	F 2		68	ПП	ПП
23	23	P X	π		69	P ПП	$x < 0$
24	24	2	2	71	71	P 7	P 7
25		F X		72	72	F 7	F 7
26	26	X	X	73		P C _x	
(30)							
31	31	P 3	P 3	74	74	7	7
32	32	F 3	F 3	75		F C _x	
33	33	P \div	e^x	76 (80)	76	C _x	C _x
34	34	3	3		78	C/П	C/П
35		F \div			79	P C/П	$x \neq 0$
36	36	\div	\div	81	81	P 8	P 8
(40)							
38		X ^y	X ^y	82	82	F 8	F 8
39		P X ^y	НОП	83	83	P -	cos
41	41	P 4	P 4	84	84	8	8

Адрес	Код	Клавиши	Оператор	Адрес	Код	Клавиши	Оператор
42	42	F 4	F 4	85		F -	
43	43	P ,	O	86 (90)	86	-	-
44	44	4	4	91	91	P 9	
45	45	F ,	1/x	92		F 9	
46	46	,	,	93	93	P +	sin
(50)							
	48	B/O	B/O	94	94	9	9
	49	P B/O	$x \geq 0$	95		F +	
51	51	P 5	P 5	96 (-0)	96	+	+

рачиваемому на один шаг после ввода очередного оператора в режиме программирования. Поэтому при вводе более шестидесяти операторов программная память переполняется и избыточные операторы замещают ранее записанные начальные операторы программы.

После ввода программы микрокалькулятор переводят оператором РР в рабочий режим и заносят исходные данные в регистры памяти и операционного стека. Для отладки программы автоматических вычислений по шагам нажимают клавишу ПП, проверяя по индикатору результаты выполнения операторов программы.

При нажатии клавиши В/О в рабочем режиме программная память "поворачивается" в начальное положение. Поэтому для пуска программы с ее первого оператора нажимают клавиши В/О и С/П, что приводит к считыванию из программной памяти кода первого оператора, после выполнения которого программная память "поворачивается" на один шаг и считывается код следующего оператора. Если считывается код оператора С/П, то выполнение программы прекращается и на индикаторе высвечивается содержимое регистра Х. Если считывается код оператора безусловного перехода БП или ПП, то выполнение программы продолжается с оператора, определяемого следующим за ПП или БП указателем перехода. Если считывается код команды проверки условия, то при его выполнении следующим считывается оператор, записанный в программе после условного оператора. Если же условие не выполнено, то управление

передается оператору, указываемому указателем перехода условного оператора.

При нажатии для пуска программы только клавиши С/П выполнение программы начнется с места ее предыдущей остановки, что может привести к ошибочным результатам вычислений. В тех случаях, когда выполнение программы должно начаться не с первого оператора, нажимают клавиши команд БП А С/П, где А – указатель перехода к оператору, с которого должно начинаться выполнение программы.

При необходимости "аварийного" прекращения выполнения программы автоматических вычислений (например, при "зацикливании" программы, когда нарушены условия выхода из цикла) достаточно нажать клавишу С/П.

Подобным образом в режиме автоматических вычислений работают и другие программируемые микрокалькуляторы, но обычно в их входных языках предусматривается передача управления с помощью меток или указателей перехода, набираемых цифрами адреса (например, в микрокалькуляторах "Электроника Б3-34" или "Электроника МК-54").

Во многих программируемых микрокалькуляторах (в частности, в микрокалькуляторах "Электроника Б3-34", "Электроника Б3-54", "Электроника МК-54" и "Электроника МК-56" с практическими совпадающими входными языками) кроме рассмотренных прямых методов адресации переходов и обращений к памяти возможна и косвенная адресация. При косвенной адресации оказывается возможным выполнять над адресами такие же операции, как и над операндами. Если прямая адресация соответствует командам типа "Отнеси это в комнату 5!", то косвенной адресации соответствуют команды типа "Узнай в комнате 5, кому это отнести!".

1.4. Искусство программирования

Оглавление этого раздела не случайно совпадает с названием многотомной монографии [6] для специалистов по программированию универсальных ЭВМ, так как алгоритмические языки таких машин отличаются от входных языков программируемых микрокалькуляторов в основном лишь способом распределения данных в памяти и вызова из нее.

Относительно большая емкость запоминающих устройств "больших" ЭВМ обычно не накладывает существенных ограничений на

длину программы, но искусство программистов в основном заключается в уменьшении числа операторов программы. Малая емкость запоминающих устройств микрокалькуляторов существенно затрудняет решение сложных задач и в этом случае при составлении программ приемлемой длины приходится творчески использовать все лексические и синтаксические особенности входного языка как программируемых, так и непрограммируемых микрокалькуляторов.

Микрокалькуляторы, как и ЭВМ любого другого типа, предназначены для ускорения вычислений. Однако при вычислениях на микрокалькуляторе значительные затраты времени связаны с выполнением вспомогательных операций и основной критерий качества (оптимальности) программы вычислений заключается в минимальных затратах времени на решение задачи, включая как ввод и выполнение программы, так и вспомогательные операции: ввод исходных данных, заполнение вычислительных бланков и т. п.

Составление качественных программ решения сложных задач связано со значительными затратами времени, которые сокращаются при сведении исходной задачи к типовым вычислениям, выполняемым по заранее составленным и отлаженным программам. Затраты времени на составление оптимальных программ решения типовых задач с избытком окупаются при повторных вычислениях по таким программам. Однако требования к форме записи составляемых и готовых программ существенно различны. При составлении программ приходится многократно переписывать их текст, изменения выбор операторов и их сочетаний. Поэтому форма записи составляемых программ должна быть предельно компактной и обеспечивать минимальные затраты времени.

В инструкциях по применению микрокалькуляторов программы вычислений с двухместными операторами обычно представляют, помещая функциональные операторы в прямоугольники, символизирующие нажимаемые клавиши, и записывая операторы набора операндов слитно, как при записи обычных чисел, например:

C 2,68 **X** 1,76 **+** 0,78 **=**

или, в общем случае, когда значения операндов заранее неизвестны:

C *a* **X** *b* **+** *c* **=**

Такая форма записи облегчает образное восприятие программы пользователем и удобна для хранения готовых программ. В этом

случае часто оказывается целесообразным указывать в квадратных скобках и высвечиваемые на индикаторе результаты вычислений. Однако при составлении программ вычерчивание символов клавиш неоправданно и целесообразно отделять операнды и операторы вертикальными линиями:

| C | a | x | b | + | c | = |

или пробелами:

C 2,68 X 1,76 + [4,7168] 0,78 = [5,4968].

Менее однозначны применяемые формы записи программ с операторами, вводимыми после нажатия префиксных клавиш. Иногда в таких записях указывают лишь символы, обозначенные на нажимаемых клавишиах. Например, вычисление арифметического выражения $y = a + \lg e^{\operatorname{arctg} b}$ на микрокалькуляторе "Электроника Б3-36" (см. рис. 4, б) в такой записи принимает вид

[C] a [+] b [arc] F [3] F [7] F [5] [=].

Подобная запись затрудняет понимание алгоритма вычислений и поэтому чаще в прямоугольниках, символизирующих клавиши, нажимаемые после префиксной, указывают операторы, обозначенные над клавишами

[C] a [+] b [arc] F [tg] F [ex] F [lg] [=]

или в упрощенной форме,

| C | a | + | b | arc | F | tg | F | ex | F | lg | = |.

Однако подобная запись, хотя и более ясно представляет алгоритм, является избыточной, так как префиксная клавиша F не связана с алгоритмом вычислений, а следующие за ней операторы не могут быть введены без нажатия этой клавиши. Поэтому в дальнейшем будем записывать в программе только операторы, представляющие алгоритм вычислений, независимо от способа их ввода:

| C | a | + | b | arc | tg | ex | lg | = |.

В то же время символы префиксных операторов, очевидно, должны быть сохранены в тех случаях, когда они являются составными частями более сложных операторов, например оператора arc, или операторов обращения к памяти и указателей передачи управления во входном языке микрокалькулятора "Электроника Б3-21".

Добавим, что в программах на алгоритмических языках "больших" ЭВМ символы префиксных клавиш также никогда не указывают (хотя печатающие устройства этих машин обычно имеют несколько рядов символов на клавишиах). Не указывают символы префиксной клавиши поднятия каретки и в обычных текстах, напечатанных на пишущих машинках.

Существенное значение имеет выбор формы записи программ автоматических вычислений и инструкций по их выполнению. В инструкции должны быть описаны размещение исходных данных и результатов вычислений в памяти и операционных регистрах, а также способ выполнения программы. Для компактной записи инструкции формулой вида $a = PN$ будем обозначать занесение исходного числа a в регистр N , а формула вида $PN = a$ будет означать, что результат вычислений a хранится в регистре N . Например, инструкция по выполнению программы на микрокалькуляторе "Электроника Б3-21" в виде записи $a = P2, b = C1, c = PX$ В/О С/П $PX = d, P3 = e, C6 = f$ будет означать, что перед пуском программы числа a, b и c следует занести соответственно в регистры 2, С1 и X , для пуска программы необходимо нажать клавиши В/О и С/П, а результаты d, e и f выполнения программы хранятся соответственно в операционном регистре X и регистрах 3 и С6 памяти.

Для длительного хранения программ автоматических вычислений обычно используют запись в виде таблицы, каждая строка которой, кроме символа команды (оператора, команды условного или безусловного перехода, указателя передачи управления), содержит дополнительную информацию — адреса шагов программы, коды команд, высвечиваемые в режиме программирования, символы нажимаемых клавиш, описания операций и иногда содержимое операционных регистров и регистров памяти.

Примером такой записи может служить табл. 2, где приведена программа автоматического вычисления вещественных корней уравнения $x^2 + a_1x + a_0 = 0$ по формуле $x_{12} = -a_1/2 \pm \sqrt{(-a_1/2)^2 - a_0}$.

Подобная табличная запись чрезмерно громоздка для ее использования в процессе составления программ, когда их текст приходится многоократно переписывать в поисках наилучшего варианта. На входном языке микрокалькулятора "Электроника Б3-21" в этих случаях удобна компактная запись программы по строкам из 12 шагов, соответствующим двум "страницам" программной памя-

ти. Приведенная в табл. 2 программа в компактной записи

$F7 \ 2 \div /-/ P4 x^2 \uparrow F8 - x \geq 0 P \div \sqrt{}$
 $\uparrow F4 \div P2 F4 XY - C/P 0 0 C/P$

занимает только две строки при той же инструкции.

Таблица 2

Вычисление вещественных корней уравнения $x^2 + a_1x + a_0 = 0$
на микрокалькуляторе "Электроника Б3-21"

№	Адрес	Код	Клавиши	Оператор	Операция	X	Y
1	01	72	F 7	F 7	Вызвать a_1 из регистра 7	a_1	0
2	02	24	2	2	Набрать цифру 2	2	a_1
3	03	36	\div	\div	Вычислить $a_1/2$	$a_1/2$	a_1
4	04	56	/-/	/-/	Принять $\alpha = -a_1/2$	α	a_1
5	05	41	P 4	P 4	Занести в регистр 4	α	a_1
6	06(10)	55	F /-/	x^2	Возвести в квадрат	α^2	a_1
7	11	06	\uparrow	\uparrow	Заслать в регистр Y	α^2	α^2
8	12	82	F 8	F 8	Вызвать a_0 из регистра 8	a_0	α^2
9	13	86	-	-	Вычислить $D = \alpha^2 - a_0$	D	α^2
10	14	49	P B/O	$x \geq 0$	Проверить $D \geq 0$	D	α^2
11	15	33	P \div	P \div	Перейти по адресу 33	D	α^2
12	16(20)	65	F ВП	$\sqrt{\quad}$	Извлечь корень	\sqrt{D}	α^2
13	21	06	\uparrow	\uparrow	Заслать в регистр Y	\sqrt{D}	\sqrt{D}

№	Адрес	Код	Клавиши	Оператор	Операция	X	Y
14	22	42	F 4	F 4	Вызвать α из регистра 4	α	\sqrt{D}
15	23	96	\div	\div	Вычислить $x_1 = \alpha + \sqrt{D}$	x_1	\sqrt{D}
16	24	21	P 2	P 2	Занести в регистр 2	x_1	\sqrt{D}
17	25	42	F 4	F 4	Вызвать α из регистра 4	α	\sqrt{D}
18	26(30)	16	XY	XY	Изменить порядок операндов	\sqrt{D}	α
19	31	86	-	-	Вычислить $x_2 = \alpha - \sqrt{D}$	x_2	α
20	32	78	C/П	C/П	Прекратить вычисления	x_2	α
21	33	04	0	0	Набрать цифру 0	0	D
22	34	04	0	0	Набрать цифру 0	00	D
23	35	78	C/П	C/П	Прекратить вычисления	00	D

Инструкция: $a_1 = P7$, $a_0 = P8$ В/О С/П РХ = x_2 , Р2 = x_1 или РХ = 00, если нет вещественных корней.

Адрес a d любого оператора в такой записи легко определить по номеру $a = 0, 1, \dots, 9$ фрагмента полустроки (учитывая, что левые полустроки имеют четные номера: $a = 0, 2, 4, 6, 8$, а правые – нечетные: $a = 1, 3, 5, 7, 9$) и порядковому номеру $b = 1, 2, \dots, 6$ шага в этой полустроке. В этой записи отсутствуют коды операторов, высвечиваемые в режиме программирования для контроля правильности их ввода, но при небольшом навыке и четком нажатии клавиш при вводе программы в программную память нет необходимости в контроле каждого оператора – достаточно проверять

лишь высвечиваемые адреса (20, 40, 60, 80, -0) крайних правых шагов в строках.

Для сокращения времени составления программ целесообразно также упростить символы сложных по начертанию операторов. Поэтому в настоящей книге операторы \odot и \ominus поворота кольцевого стека заменены символами \rightarrow и \leftarrow соответственно, а для оператора \overrightarrow{XY} использована упрощенная запись XY .

Однако, как ни существенна форма записи программ, их качество определяется затратами времени на решение задачи. Уменьшение длины программы в процессе ее составления приводит к уменьшению затрат времени на ввод программы как в связи с уменьшением количества нажимаемых клавиш, так и в связи с уменьшением вероятности ошибок. Однако сокращение длины программы за счет увеличения обращений к памяти и особенно к вычислительному бланку при хранении в нем результатов промежуточных вычислений недопустимо, так как в этом случае резко возрастают затраты времени на решение задачи.

Значительные затраты времени связаны с набором многозначных операндов. Поэтому следует по возможности полнее использовать регистры памяти и операционного стека в соответствии с лексическими и синтаксическими особенностями входного языка микрокалькулятора, чтобы уменьшить число повторных вводов многозначных операндов. Например, многократное вычисление арифметического выражения $y = a/x^2 + x$ для различных значений аргумента x на микрокалькуляторе "Электроника Б3-23" (см. рис. 4, а) можно непосредственно выполнять по программе

С $a \div x \div x + x =$,
трижды набирая значения операнда x при каждом выполнении программы. Между тем, воспользовавшись правилами работы операционного стека этого микрокалькулятора (см. рис. 6), вычисления можно выполнять по программе

С $x \div = = X a + x =$,

требующей только двукратного набора операнда x .

При вычислениях по той же формуле на микрокалькуляторе "Электроника Б3-36" (см. рис. 4, б) целесообразно занести значение a в регистр памяти и вычислять результаты по программе
 $x + (\div = = X ИП) =$.

Несмотря на несколько непривычный вид такой программы, составленной в соответствии с синтаксическими правилами работы операционного стека (см. рис. 7), она позволяет при каждом вычислении заданного арифметического выражения набирать только новое значение операнда x .

При составлении программ автоматических вычислений требование минимального числа операторов может оказаться первоочередным, так как решение задачи с помощью нескольких последовательно выполняемых программ связано с существенным увеличением времени вычислений. Поэтому следует использовать все особенности входного языка программируемого микрокалькулятора для решения задачи с помощью одной программы.

Основными средствами сокращения длины программы с повторяющимися фрагментами являются обращения к подпрограммам и организация итерационных циклов. Если программа содержит n одинаковых фрагментов из m шагов каждый, то выделение одного такого фрагмента в подпрограмму приводит к сокращению длины программы при условии $mn > m + 1 + 2n$, так как подпрограмма из m шагов дополняется оператором В/О, а на месте каждого фрагмента исходной программы, вынесенного в подпрограмму, записывается оператор ПП A , занимающий два шага, где A – указатель передачи управления оператору подпрограммы.

Если программа содержит последовательность из n одинаковых фрагментов с m шагами каждый, то можно заменить ее циклом, в котором один такой фрагмент выполняется n раз. Для этого повторяемый фрагмент дополняют счетчиком итераций и условным оператором, обеспечивающим выход из цикла после n выполнений фрагмента. На входном языке микрокалькулятора "Электроника Б3-21" для выполнения n раз фрагмента M записывается $M FN 1 - PN x = 0 A$, где число повторений вычислений по фрагменту M (число итераций) n предварительно заносят в регистр N памяти. Так как вспомогательные операторы занимают 6 шагов, то длина программы при организации цикла уменьшается при соблюдении условия $mn > m + 6$. Если для временного хранения результатов вычислений по фрагменту M приходится дополнительно вводить в программу операторы обращения к памяти, то это неравенство следует соответственно изменить.

Общие приемы сокращения длины программы связаны с предварительным преобразованием расчетных соотношений и использова-

нием лексических и синтаксических особенностей входного языка. Например, вычисление выражения $\ln((x+1)/(x-1))$ на входном языке микрокалькулятора "Электроника Б3-21" при $x = P2 X$ можно непосредственно отобразить фрагментом

P2 1 + P3 F2 1 - ↑ F3 XY ÷ ln.

Длину этого фрагмента можно несколько сократить, изменив порядок операций:

P2 1 - P3 F2 1 + ↑ F3 ÷ ln.

Однако если учесть сохранение содержимого операционного регистра после выполнения двухместной операции и засылку содержимого регистра X при наборе числа, то этот фрагмент можно сократить еще больше:

P2 1 + XY 2 - ÷ ln.

В некоторых случаях длину программы удается сократить и при рациональном использовании лексических особенностей входного языка. Например, при вычислении выражения $(\sin x)a$ на микрокалькуляторе "Электроника Б3-21" вместо фрагмента $\sin \uparrow F2 X$ при $a = P2$, $x = P2 X$ целесообразно использовать фрагмент $e^{jx} F2 X$. Программы автоматических вычислений на входном языке микрокалькуляторов этого типа часто удается сократить и при рациональном использовании кольцевого стека памяти.

При большом числе исходных данных и результатов вычислений (включая те промежуточные результаты, которые приходится временно хранить в памяти) задачи часто решают последовательным выполнением нескольких программ автоматических вычислений с меньшим количеством операндов. Значительных потерь времени на решение задачи в подобных случаях часто удается избежать, рационально используя одни и те же регистры памяти в качестве буферных регистров для временного хранения различных промежуточных результатов и нормируя арифметические выражения для уменьшения числа исходных данных, а также записывая часть исходных данных непосредственно в программу. При этом запись однозначных операндов в программу не увеличивает ее длину, а для микрокалькулятора "Электроника Б3-21" даже уменьшает. При записи в программу многозначных операндов для уменьшения длины про-

граммы целесообразно записывать, например, $/3/$ вместо $|1|0|$, $|3|3|3| \dots$, или $|2|\sqrt{}$ вместо $|1|1|$, $|4|1| \dots$, или $|3|4|x^2|$ вместо $|1|1|5|6|$ и т. п.

Время решения задач с помощью микрокалькуляторов может быть сокращено и при использовании программ (особенно для автоматических вычислений), составленных на входных языках микрокалькуляторов других типов. В этом случае возникает проблема перевода программ на входной язык микрокалькулятора используемого типа. Подобный перевод наиболее прост и выполняется "дословно" при совпадении в пределах переводимой программы словарного запаса и синтаксических правил обоих входных языков. Однако при существенных отличиях в синтаксисе и лексике входного языка может оказаться целесообразным составление по программе-оригиналу описания или схемы алгоритма, по которым и составляют программу-перевод.

Во всех случаях перевод программы упрощается, если язык программы-перевода отличается более бедным словарным запасом и простотой синтаксиса. Поэтому в настоящей книге программы автоматических вычислений приведены на наиболее простом входном языке микрокалькулятора "Электроника Б3-21", с которого их несложно перевести на входной язык любого другого программируемого микрокалькулятора (например, "Электроника Б3-34") или настольных и стационарных ЭВМ.

1.5. Будьте внимательны!

Существенным этапом решения задачи является проверка точности полученных результатов. Причиной их погрешностей могут быть грубые ошибки, допущенные при составлении исходной математической модели, выборе алгоритма и выполнении программы вычислений. Однако даже при отсутствии таких грубых ошибок результат вычислений на микрокалькуляторе обычно оказывается приближенным. Поэтому точность результата a вычислений обычно оценивают предельными абсолютной Δa и относительной $\delta a = \Delta a / |a|$ погрешностями. Их выбирают на основании имеющейся информации так, чтобы точное значение результата находилось по возможности ближе к границе $a \pm \Delta a = a(1 \pm \delta a)$ интервала со средним значением a .

Погрешность результата a часто удобно оценивать по его десятичному представлению. Верными называют первые слева n значащих цифр числа a , если его абсолютная погрешность не превышает половины единицы n -го разряда ($\Delta a \leq 5 \cdot 10^{-n}$), и верными в широком смысле, если его абсолютная погрешность не превышает половины единицы n -го разряда ($\Delta a \leq 10^{-n+1}$).

Отклонение результата вычислений на микрокалькуляторе от его точного значения связано с методическими погрешностями, зависящими от метода вычислений, и операционными погрешностями, возникающими при выполнении отдельных операций. Операционные погрешности связаны с ограниченным числом разрядов в машинных представлениях операндов. При вычислениях с ограниченной разрядностью результат, по абсолютной величине меньший A_{\min} или больший A_{\max} , попадает в область машинных нуля или бесконечности. Однако даже при попадании результата в область представления чисел на индикаторе с r цифровыми разрядами число значащих цифр мантиссы не превышает r и, следовательно, результат не может содержать более r верных цифр. Округление результата вычислений до r значащих цифр в микрокалькуляторах выполняется различными способами, среди которых наиболее простыми являются округление отбрасыванием и округление по дополнению.

При округлении отбрасыванием сохраняются первые r значащих цифр мантиссы числа независимо от величины ее отбрасываемой части. Абсолютная погрешность округления в этом случае не превышает значения единицы r -го разряда ($\Delta \leq 10^{1-r}$). При округлении по дополнению r -я значащая цифра мантиссы сохраняется без изменений, если отбрасываемая ее часть не превышает половины единицы r -го разряда, и увеличивается на единицу в противном случае. В этом случае погрешность округления меньше и не превышает половины единицы r -го разряда ($\Delta \leq 5 \cdot 10^{-r}$).

При выполнении последовательности операций погрешности округлений накапливаются и полная погрешность окончательного результата может оказаться значительной. В некоторых микрокалькуляторах (например, "Электроника С3-15" и "Электроника Б3-19М") вычисления выполняются над операндами с большим числом разрядов, чем высвечиваемые на индикаторе результаты вычислений. Это существенно снижает полную погрешность результата последовательности операций, высвечиваемого на индикаторе.

Методические погрешности возникают при вычислениях с помощью приближенных методов, когда точный результат теоретически достигается при бесконечном числе последовательных приближений. Практически выполнимо лишь конечное число операций, в связи с чем и возникает методическая или остаточная погрешность результата. При вычислениях с помощью карандаша и бумаги методическую погрешность уменьшают, увеличивая число операций и разрядность операндов. Однако при вычислениях на микрокалькуляторе число значащих цифр и, следовательно, достижимая точность результата ограничены рабочей разрядностью *r* индикатора.

Микропрограммы вычисления элементарных функций в ПЗУ микрокалькулятора обычно составлены с использованием методов последовательных приближений. Однако погрешность вычисления этих функций не может быть уменьшена и ее следует рассматривать как операционную. Предельные значения погрешностей обычно указаны в паспортных данных микрокалькулятора.

Ограниченнaя разрядность операндов при вычислениях на микрокалькуляторах может явиться причиной серьезных ошибок, что можно показать на следующем примере. При вычислении на восьмиразрядном микрокалькуляторе "Электроника Б3-18А" арифметического выражения $25\ 000 \times 10\ 000 - 24\ 995 \times 10\ 000 =$ возникает переполнение, тогда как после приведения того же арифметического выражения к виду $(25\ 000 - 24\ 995) \times 10\ 000 = 50\ 000$ результат оказывается правильным.

При умножении чисел

$$0,0002 \times 0,0004 \times 4000 \times 25\ 000$$

на индикаторе высвечивается нуль; при перестановке сомножителей $4000 \times 25\ 000 \times 0,0002 \times 0,0004$

возникало переполнение и лишь при перестановке

$$4000 \times 0,0002 \times 0,0004 \times 25\ 000 = 8$$

получался правильный результат.

Необходимо отметить, что сочетательный и распределительный законы справедливы лишь для чисел с неограниченной разрядностью. В приведенных примерах результаты промежуточных операций попадают в область машинного нуля или бесконечности, что и приводит к ошибочным результатам.

При расчетах на микрокалькуляторах, как и на любых других

цифровых ЭВМ, необходимо помнить о том, что при различиях в порядке слагаемых более чем на величину рабочей разрядности r меньшее число становится машинным нулем относительно большего числа и полностью округляется. Если же начать сложение с меньших по порядку слагаемых, то в случае попадания результата их сложения в разрядную "сетку" микрокалькулятора числа в старших разрядах этого результата складываются с цифрами в младших разрядах большего слагаемого и влияют на конечную сумму.

Поэтому были разработаны правила сложения и умножения, которые обеспечивают пользователям микрокалькуляторов получение суммы и произведения на микрокалькуляторе любого типа с минимальными погрешностями:

сложение следует начинать с меньших по порядку слагаемых;
для вычисления произведения следует представить сомножители в стандартной показательной форме, умножить ихmantиссы и сложить порядки.

При вычислениях на микрокалькуляторе может сложиться мнение, что нет необходимости контролировать погрешности, так как даже при пяти верных цифрах результата его относительная погрешность меньше 0,05%, тогда как практическая точность измерения физических величин обычно не превышает десятых долей процента. Зададим себе вопрос: "На сколько увеличится объем Земли, если считать ее шаром с радиусом 6700 км, при увеличении этого радиуса на один сантиметр?" На первый взгляд искомое приращение будет пренебрежимо малым, так как приращение радиуса составляет менее $1,5 \cdot 10^{-6}\%$. Составим расчетную формулу

$$\Delta V = ((R + \Delta R)^3 - R^3) 2\pi/3,$$

чтобы убедиться в невозможности вычисления приращения объема на микрокалькуляторе: даже при показательной форме представления чисел результат вычислений по этой формуле оказывается равным нулю.

Если преобразовать формулу, разложив куб суммы и сократив подобные члены, то получим искомое увеличение объема $\Delta V = (3R^2 \Delta R + 3R\Delta^2 R + \Delta^3 R) 2\pi/3 \approx (R + \Delta R) 2R\Delta R\pi \approx 2\pi R^2 \Delta R = 2820,5 \text{ км}^3$ с погрешностью $\Delta \Delta V \approx 2\pi \Delta R/3 \approx 2,1 \cdot 10^{-15} \text{ км}^3$.

Таким образом, увеличение радиуса Земли всего на 1 см приводит к увеличению ее объема на огромную величину.

Читатель может заинтересоваться способами оценки полной опе-

рациональной погрешности результата вычислений. Обычно результат вычислений рассматривают как функцию операндов $x(a_1, \dots, a_n)$. При малых погрешностях операндов это позволяет оценивать предельную абсолютную погрешность результата по формуле

$$\Delta x = \sum_{i=1}^n K_i \Delta a_i = \sum_{i=1}^n |\partial x / \partial a_i| \Delta a_i,$$

где модули частных производных K_i можно рассматривать как коэффициенты чувствительности результата к абсолютным погрешностям операндов. По этой формуле несложно оценить коэффициенты: $K_a = K_b = 1$ для суммы $x = a + b$ и разности $x = a - b$; $K_a = K_b = a$ для произведения $x = ab$; $K_a = 1/b$ и $K_b = a/b^2$ для частного $x = a/b$; $K_a = 1/2\sqrt{a}$ для корня $x = \sqrt{a}$; $K_a = 2a$ для квадрата $x = a^2$; $K_a = 1/a$ для логарифма $x = \ln a$ и т. п.

Полную погрешность результата нескольких операций удобно определять с помощью графа, вершины-источники которого соответствуют погрешностям Δa_i исходных данных и Δ_k результатов промежуточных операций, вершина-сток – полной погрешности Δx результата вычислений. Промежуточные вершины для наглядности обозначают символами операторов. Ветвям, направленным от операндов к вершинам-операторам, приписывают веса, равные коэффициентам чувствительности, и полную погрешность Δx находят по графу в соответствии с приведенной выше формулой.

Например, погрешность результата $x = (a + b)^2 - a^2$ определяют по графу, показанному на рис. 10, а, согласно формуле $\Delta x = \Delta_3 + \Delta_4 + 2a \Delta a + \Delta_2 + 2(a + b)(\Delta_1 + \Delta a + \Delta b)$, и при $a = 20$,

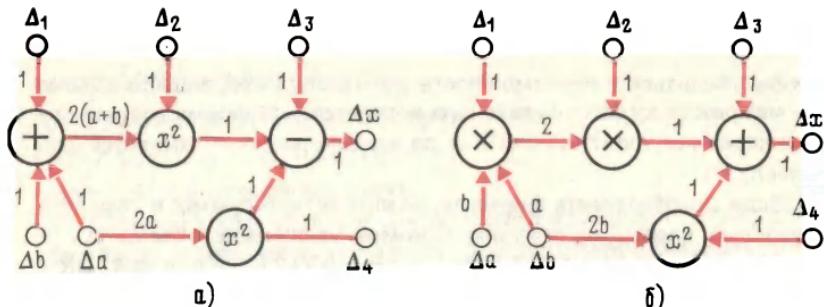


Рис. 10. Графы накопления погрешностей при вычислении $x = (a + b)^2 - a^2$

$b = 1$, $\Delta a = \Delta b = 10^{-5}$, $\Delta_1 = \Delta_2 = \Delta_3 = \Delta_4 = 10^{-7}$ погрешность результата вычислений $\Delta_x = 1,2445 \cdot 10^{-3}$, что значительно превышает погрешности исходных данных и округления результатов операций.

Среди простейших операций наибольшей погрешностью отличается вычисление близких чисел, так как относительная погрешность результата $\delta_x = \Delta_x / |x| = (\Delta a + \Delta b) / (a - b)$ быстро возрастает при уменьшении абсолютного значения результата. Для уменьшения погрешности результата в этих случаях используют различные приемы замены вычитания сложением, например:

$$(a+b)^2 - a^2 = 2ab + b^2 ;$$

$$(a+b)^3 - a^3 = 3a^2b + 3ab^2 + b^3 ;$$

$$\sqrt{a+b} - \sqrt{a} = b / (\sqrt{a+b} + \sqrt{a}) \text{ и т. п.}$$

При использовании первого из преобразованных выражений полная погрешность результата определяется с помощью графа (рис. 10, б) как $\Delta_x = \Delta_3 + \Delta_4 + 2b\Delta b + \Delta_2 + 2(\Delta_1 + b\Delta a + a\Delta b)$. При $a = 20$, $b = 1$, $\Delta a = \Delta b = 10^{-5}$, $\Delta_1 = \Delta_2 = \Delta_3 = \Delta_4 = 10^{-7}$ эта погрешность $\Delta_x = 4,405 \cdot 10^{-4}$, или почти в три раза меньше погрешности вычислений по схеме, отображенной графиком на рис. 10, а.

При необходимости на микрокалькуляторе можно получать и точные результаты арифметических операций с числом значащих цифр, превышающим разрядность микрокалькулятора. Так, для точного сложения многозначных операндов их следует уравнять по порядку мантиссы, разбить на блоки, начиная с младших разрядов, содержащие по $r - 1$ цифр (старший блок может содержать меньшее число цифр) и суммировать на микрокалькуляторе блоки одинаковых порядков. Например, для сложения чисел 59585 , 3627218 и $2,3689781565 \cdot 10^2$ их следует привести к целочисленным мантиссам и сложить по блокам:

595853	6272180
+	+
2368	9781565
598221	16053745
598222	$6053745 \cdot 10^{-8}$

Сложение нескольких многозначных операндов на программируемом микрокалькуляторе целесообразно выполнять в режиме автоматических вычислений, воспользовавшись, например, следующей программой для микрокалькулятора "Электроника Б3-21".

Программа точного сложения чисел, содержащих до 15 значащих цифр:

F8	1	ВП	7	-	↑	F5	+	x < 0	X	1	ВП
7	+	P8	БП	F ÷ P8	F7	1	+	P7	F7	↑	
F4	+	P7	C/P	БП	P0						

Перед выполнением этой программы старший и младший блоки первого операнда заносят в регистры 7 и 8, следующего операнда – в регистры 4 и 5 соответственно. После выполнения программы старший блок результата хранится в регистрах X и 7, а младший – в регистре 8. Следующие операнды заносят по блокам в регистры 4 и 5 и нажимают только клавишу C/P. При $595853 = P7, 6272180 = P8, 2368 = P4, 9781565 = P5$ получим $RX = P7 = 598222, P8 = 6053745$.

Для точного умножения на микрокалькуляторе с восьмиразрядной мантиссой многозначные операнды уравнивают по порядку, приводя их мантиссы к целым числам и разбивая последние, начиная с младших разрядов, на блоки a_i из четырех (или менее в старшем блоке) цифр. В этом случае операнды a и b , содержащие до восьми или восемь значащих цифр, могут быть представлены выражениями $a = a_2 \cdot 10^4 + a_1$ и $b = b_2 \cdot 10^4 + b_1$, при умножении которых имеем: $ab = a_2 b_2 \cdot 10^8 + (a_1 b_2 + a_2 b_1) \cdot 10^4 + a_1 b_1$.

Следовательно, на микрокалькуляторе можно получить точный результат, вычислив произведения $a_i b_j$ и сложив их со сдвигом на четыре разряда.

Для вычисления произведения чисел, например $5,213348 \cdot 10^8$ и $3,4228195 \cdot 10^{-2}$ с сохранением всех цифр результата, их мантиссы приводят к целым числам $521\ 3348 \cdot 10^2$ и $3422\ 8195 \cdot 10^{-9}$ и разбивают на блоки $a_2 = 521, a_1 = 3348, b_2 = 3422, b_1 = 8195$. Тогда порядок произведения равен сумме порядков сомножителей, а его мантисса определяется как

$$\begin{array}{r} a_2 b_2 \quad 178\ 2862 \\ a_1 b_2 \quad + \quad 1145\ 6856 \\ a_2 b_1 \quad + \quad 426\ 9595 \\ a_1 b_1 \quad + \quad 2743\ 6860 \\ \hline 178\ 4434\ 9194\ 6860 \cdot 10^{-7}. \end{array}$$

При необходимости точное умножение нескольких операндов также можно автоматизировать с помощью программируемого микрокалькулятора.

Уточнение дробного результата деления $x = a/b$ с числом значащих цифр, большой разрядности индикатора, можно свести¹ к вычислению поправки результата $\Delta x = (a - bx) : b$. Например, при делении 2 на 7 на индикаторе с показательной формой представления чисел получим $2,8571428 \cdot 10^{-1}$. Точное значение произведения $bx = 2,8571428 \cdot 10^{-1} \times 7 = 1,9999996$ и, следовательно, $\Delta x = (2 - 1,9999996) : 7 = 4 \cdot 10^{-8} : 7 = 0,57142857 \cdot 10^{-8}$, откуда $x = 0,2857142857142857$.

Однако подобные точные вычисления можно выполнять, естественно, лишь в исключительных случаях, когда это оправдывает затраты времени. Обычно же при вычислении достаточно хотя бы грубо оценивать погрешности результата и, при необходимости, выбирать алгоритмы вычислений, обеспечивающие вычисления с требуемой точностью. Особую бдительность следует проявлять в тех случаях, когда промежуточный результат может попасть в область машинной бесконечности или, еще опаснее (так как при этом не возникает сигнала, подобного извещающему о переполнении), в область машинного нуля, что приведет к ошибке вычислений. Однако и малые погрешности, накапливающиеся в процессе вычислений, могут привести к большим ошибкам результата вычислений.

Кстати, осваивая микрокалькулятор, можно попасть в подобную ситуацию, получив при вычислении арифметического выражения $((10 : 3 - 4) 6 + 4) 10^8$ в ответе -20 , хотя проверка с помощью карандаша и бумаги дает правильный нулевой ответ. Причина расходжения заключалась в погрешности округления отбрасыванием результата операции деления. При сокращении общего знаменателя она устраняется.

¹ На микрокалькуляторе, округляющем отбрасыванием результата вычитания.

2
ГЛАВА

**С МИКРОКАЛЬКУЛЯТОРОМ СКВОЗЬ "ДЕБРИ"
МАТЕМАТИКИ**

**КАК ОБОЙТИСЬ БЕЗ ТАБЛИЦ?
В ПОИСКАХ КОРНЯ
СКОЛЬКО БЫЛО ВИНОГРАДА, ИЛИ КАК РЕШИТЬ СИСТЕМУ
УРАВНЕНИЙ?
НЕМНОГО СТАТИСТИКИ
БЛУЖДАНИЕ В ТУМАНЕ, ИЛИ ОПТИМИЗАЦИЯ
ПАРАМЕТРОВ**

С МИКРОКАЛЬКУЛЯТОРОМ СКВОЗЬ "ДЕБРИ" МАТЕМАТИКИ

2.1. Как обойтись без таблиц?

До изобретения ЭВМ широкое распространение нашли таблицы значений наиболее часто встречающихся на практике функций, подобные известным каждому школьнику таблицам логарифмов. Однако владелец даже простейшего микрокалькулятора не нуждается в таких таблицах и при достаточном усердии может вычислить любую функцию с высокой точностью. В тех случаях, когда входной язык используемого микрокалькулятора не содержит оператора вычисления нужной функции, необходимо предварительно составить выражение, приближающее (*аппроксимирующее*) значение этой функции в заданном интервале аргумента.

В качестве аппроксимирующих выражений часто используют степенные многочлены

$$P(x) = \sum_{i=0}^n a_i x^i = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n,$$

коэффициенты и число членов которых подбирают в зависимости от требуемой точности вычисления аппроксимируемой функции.

Попытаемся составить такой многочлен для приближенного вычисления функции $f(x) = \sin x$. Так как эта функция нечетная и $\sin x = -\sin(-x)$, то коэффициенты многочлена при четных степенях аргумента должны быть равными нулю и, следовательно, первым будет член $a_1 x$. График этого члена представляет собой прямую линию, проходящую через начало координат с наклоном, определяемым значением a_1 . Этот график отображает функцию $\sin x$ в окрестностях точки $x = 0$, если он совпадает с касательной к функции $\sin x$ при $x = 0$. Так как касательная к графику функции имеет наклон, определяемый производной функции в точке касания, то $a_1 = \sin' x$ и при $x = 0$ получим $a_1 = 1$ или $\sin x \approx x$. Однако при увеличении аргумента аб-

солютная погрешность $\sin x - x$ нашего приближения будет сильно возрастать (рис. 11) и для ее компенсации потребуется учесть дополнительные члены.

Построение многочлена для вычисления функции $\sin x$ можно упростить, воспользовавшись разработанными математиками методами аппроксимации функций. Так как функция $\sin x$ дифференцируема, то при $x = x_0$ ее можно представить рядом Тейлора

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!} (x - x_0) + \frac{f''(x_0)}{2!} (x - x_0)^2 + \\ + \frac{f'''(x_0)}{3!} (x - x_0)^3 + \dots,$$

который при $x_0 = 0$ называют рядом Маклорена

$$f(x) = f(0) + f'(0)x/1! + f''(0)x^2/2! + f'''(0)x^3/3! + \\ + f^{IV}(0)x^4/4! + \dots,$$

где факториал $1! = 1$.

Определив $\sin 0 = 0$, $\sin' x = \cos x$, $\sin'' x = -\sin x$, $\sin''' x = -\cos x$, ..., получим разложение рассматриваемой функции

$$\sin x = x - x^3/3! + x^5/5! + x^7/7! + \dots = \sum_{k=0}^{\infty} (-1)^k x^{2k+1}/(2k+1)!$$

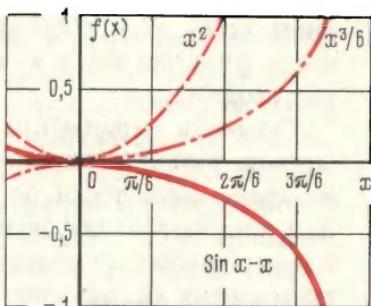
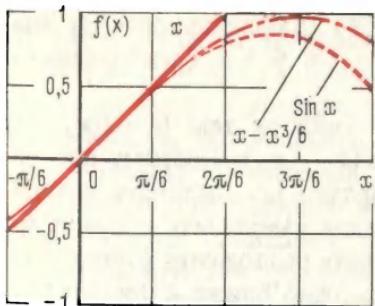


Рис. 11. Аппроксимации функции $\sin x$

Если сохранить в таком ряде только первые n членов, то получим степенной многочлен, аппроксимирующий функцию $\sin x$ с абсолютной погрешностью, не превышающей величины первого из отброшенных членов ряда $(-1)^{n+1}x^{2n+3}/(2n+3)!$. По величине этого члена можно оценить и число членов ряда, требуемых для аппроксимации функции $\sin x$ с заданной точностью.

Заметим, что для разложения в ряд Тейлора функция должна иметь все (от первого до бесконечного порядков) производные (такие функции называют аналитическими), а вычисления функций с помощью степенных многочленов, полученных подобным образом, имеет смысл выполнять только в том случае, если ряд Тейлора сходится (модуль его членов уменьшается с увеличением номеров k). Сходимость функциональных рядов, каким является ряд Тейлора, зависит от значения аргумента x . Например, ряд, получающийся при разложении функции $\sin x$, сходится при любых значениях x , в то время как для разложения

$$\operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \dots = \sum_{k=0}^{\infty} \frac{-x^{(2k+1)}}{(2k+1)} (-1)^k$$

условия сходимости выполняются только в интервале $|x| < 1$. Поэтому, чтобы вычислить значение функции $\operatorname{arctg} x$ при $|x| > 1$ следует воспользоваться соотношением $\operatorname{arctg} x = \pi/2 - \operatorname{arctg}(1/x)$, вычислив сначала значение функции $\operatorname{arctg} z$, где $z = 1/x < 1$ и, следовательно, ряд сходится.

Скорость сходимости ряда Тейлора тем больше, чем меньше значение его аргумента ($x - x_0$). Поэтому для ускорения вычислений целесообразно использовать различные преобразования, позволяющие уменьшить значение аргумента, либо постараться получить разложение относительно значения x_0 , лежащего как можно ближе к представляющему интерес значению аргумента. Примерами подобных

преобразований являются следующие формулы:

$$\sin x = \cos(x - \pi/2) = -\sin[x + (2k+1)\pi] = \sin(x + 2\pi k);$$

$$\cos x = \sin(x + \pi/2) = -\cos[x + (2k+1)\pi] = \cos(x + 2\pi k),$$

позволяющие "привести" аргумент функций $\sin x$ и $\cos x$ к интервалу $x \in [0; \pi/4]$, или представление показательной функции в виде двух сомножителей: $e^x = e^{x_1} e^{x_2}$, показатель одного из которых выбирается целым числом, и поэтому, зная $e \approx 2,71828182846$, его несложно вычислить возведением в целую степень на микрокалькуляторе любого типа, а второго — со значением x_2 , ограниченным интервалом $|x_2| \leq 0,5$, что позволит достаточно быстро достичь требуемой точности при вычислении этого сомножителя суммированием ряда

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{k=0}^{\infty} x^k/k!.$$

Методическая погрешность вычисления функции по ее разложению в ряд Тейлора может быть сделана сколь угодно малой. Для этого достаточно учесть соответствующее число членов ряда (если, конечно, ряд сходится). Однако при фиксированном числе членов ряда удаление от точки x_0 , относительно которой выполнено разложение, ведет к монотонному росту погрешности. Поэтому при вычислении значения функции по ее разложению в степенной ряд, состоящий из заранее оговоренного числа членов, напрашивается мысль попытаться изменить коэффициенты ограниченного усеченного ряда, чтобы "распределить" методическую погрешность более равномерно по заданному интервалу изменения аргумента. Для этого, например, можно так выбрать коэффициенты a_i , чтобы обеспечить точное совпадение приближенного выражения с аппроксимируемой функцией в выбранном числе точек, называемых узлами интерполяции. В общем случае для степенного многочлена n -й степени, подставляя в него

значения аргумента и функции в каждом узле интерполяции

$$f(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 + a_n x_1^n,$$

$$f(x_2) = a_0 + a_1 x_2 + a_2 x_2^2 + a_n x_2^n,$$

$$f(x_{n+1}) = a_0 + a_1 x_{n+1} + a_2 x_{n+1}^2 + a_n x_{n+1}^n,$$

можно составить систему уравнений относительно неизвестных коэффициентов a_i , которая при числе уравнений $n+1$ (и, следовательно, при выборе $n+1$ -го узла) будет иметь единственное решение. Такой способ построения интерполирующего многочлена называют методом неопределенных коэффициентов.

При равноотстоящих узлах можно обойтись и без решения составленной системы, если воспользоваться, например, готовой интерполяционной формулой Ньютона [3,7]. В промежутках между узлами интерполяции значения функции отличаются от значения интерполирующего многочлена, причем, как правило, при равноотстоящих узлах максимальные значения отклонений между различными парами смежных узлов оказываются различными (по модулю). Всегда можно так подобрать положение узлов интерполяции, чтобы модули максимальных значений погрешностей между каждой парой узлов (уже неравноотстоящих!) оказались равными. В этом случае будет достигнута максимальная точность аппроксимации на заданном интервале аргумента функции $f(x)$ многочленам выбранной степени.

В качестве примера рассмотрим аппроксимацию функции e^x многочленом второй степени $e^x \approx a_0 + a_1 x + a_2 x^2$ в интервале $x \in [0; 1]$. При аппроксимации рядом Тейлора получим $e^x \approx 1 + x + x^2/2$ с максимальной погрешностью при $x = 1$, примерно равной $\Delta_{\max} \approx 0,22$ (рис. 12, кривая I). Однако, оставив $a_0 = a_1 = 1$, определим a_2 из условия $e^1 = 2,718218 = 1 + 1 + a_2$. Если получающееся значение $a_2 = 0,718281\dots$ округлить до значения $a_2 = 0,7$, то уз-

лом интерполяции окажется значение аргумента, несколько меньшее единицы ($x \approx 0,91$), что приведет, во-первых, к более удобной формуле, а во-вторых, к более равномерному распределению погрешности по интервалу и уменьшению ее максимального (по модулю) значения до 0,03 (рис. 12, кривая 2).

Выбрав в качестве узлов интерполяции значения $x = 0$; $x = 0,5$ и $x = 1$, из системы уравнений

$$\begin{aligned} 1 &= a_0 \\ e^{1/2} &\equiv 1,64872 = a_0 + a_1/2 + a_2/4, \\ e &\equiv 2,718281 = a_0 + a_1 + a_2, \end{aligned}$$

получим: $a_0 = 1$; $a_1 = 0,8766$; $a_2 = 0,84168$. Аппроксимирующий многочлен $e^x \approx 1 + 0,8766x + 0,8466x^2$ с этими коэффициентами обеспечивает предельную погрешность $\Delta_{\max} \approx 0,0145$, достигаемую между вторым и третьим узлами интерполяции. Смешая узлы интерполяции, можно подобрать такое их положение, при котором максимальные значения модулей ошибки между началом интервала и первым узлом, первым и вторым, вторым и третьим, третьим и верхней границей интервала окажутся равными (методика определения коэффициентов с помощью

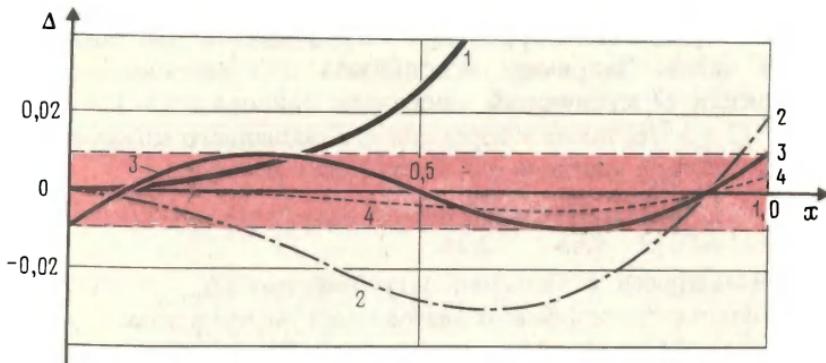


Рис. 12. Аппроксимации функции e^x

микрокалькулятора в этом случае описана в [7]). Полученный таким образом квадратичный многочлен $e^x = 1,008756 + 0,85473x + 0,84603x^2$ обеспечивает минимально возможную погрешность, не превышающую значения $\Delta < 0,0088$ (кривая 3 на рис. 12 изображает зависимость погрешности от значения аргумента для этого случая).

Многочлены, подобные последнему (только более точные), широко используются при вычислении различных функций с помощью ЭВМ, так как по хранимым в ее памяти коэффициентам вычисление требуемой функции обеспечивается небольшим числом операций. Однако эти формулы малопригодны для вычисления на микрокалькуляторах, поскольку ручной ввод многоразрядных коэффициентов утомителен, требует повышенного внимания и по времени эквивалентен выполнению нескольких операций. Практика показывает, что для получения необходимой точности проще несколько увеличить число выполняемых операций, "удлинив" аппроксимирующее выражение, но сохранить его коэффициенты немногоразрядными числами. Для получения подобных аппроксимирующих многочленов за основу целесообразно взять степенной многочлен Тейлора и попытаться уменьшить его погрешность коррекцией коэффициента при последнем члене. Например, использовав для аппроксимации функции e^x кубический многочлен Тейлора $e^x = 1 + x + x^2/2 + x^3/6$, после коррекции его последнего коэффициента получим удобную для вычислений формулу

$$e^x \approx 1 + x + \frac{x^2}{2} + \frac{x^3}{4,68} = \left(\left(\frac{x}{2,34} + 1 \right) \cdot x + 1 \right) \cdot x + 1,$$

отличающуюся и меньшей погрешностью ($\Delta_{\max} = 0,0046$, а зависимость ошибки от значения аргумента в этом случае изображена на рис. 12 в виде кривой 4). Заметим, что без коррекции последнего члена погрешность достигла бы значения $\Delta_{\max} = 0,05$, т. е. примерно в 10 раз большего.

С другой стороны, для обеспечения погрешности, не превышающей 0,005, непосредственно по некорректированному ряду Тейлора потребовало бы вычисления значений еще двух членов ряда.

Таблица 3

Вычисление элементарных функций с помощью арифметических операций

Функция	Расчетная формула	Погрешность
$\sin x$	$(x^2/12,052 - 1)^2 x$ $(((-x^2/42,25 + 1)x^2 : 20 - 1)x^2 : 6 + 1)x$ $(((-x^2/43,36 + 1)x^2 : 20 - 1)x^2 : 6 + 1)x$	$\Delta < 5 \cdot 10^{-5}$ при $ x < \pi/4$ $\Delta < 1 \cdot 10^{-7}$ при $ x < \pi/4$ $\Delta < 1,1 \cdot 10^{-5}$ при $ x < \pi/2$
$\cos x$	$(x^2 : 24,17 - 1)^2 (-x) : 2 + 1$ $((-x^2 : 30,31 + 1)x^2 : 24 - 1)x^2 : 2 + 1$ $((-x^2 : 31,24 + 1)x^2 : 24 - 1)x^2 + 1$	$\Delta < 1,4 \cdot 10^{-5}$ при $ x < \pi/4$ $\Delta < 1 \cdot 10^{-7}$ при $ x < \pi/4$ $\Delta < 1,7 \cdot 10^{-7}$ при $ x < \pi/2$
$\operatorname{tg} x$	$((((x^2 : 1,42 + 1)x^2 : 1,38 + 2)x : 5 + 1)x$ $\times x^2 : 3 + 1)x$ $x : (1 + x^2 : (3 - x^2 : 4,917))$	$\Delta < 1,2 \cdot 10^{-5}$ при $ x < \pi/4$ $\Delta < 1,7 \cdot 10^{-7}$ при $ x < \pi/2$
$\alpha = \operatorname{arctg} x$	$((0,3x^2 - 1)0,3x + 1)57x$	$\Delta < 0,15^\circ$ при $ x < 1$
$\rho = \operatorname{arctg} x$	$x : (1 + x^2 : (3 + 4x^2 : (5 + 1,08x^2)))$	$\Delta < 10^{-4}$ при $ x < 1$
e^x	$-1 : (-1 + 2x : (2 + x + x^2 : (6 + x^2 : 10,07)))$	$\Delta < 2,5 \cdot 10^{-6}$ при $ x < 1$
$\lg(1+x)$	$0,4343x : (1 + x : (2 + x : (3 + x : (1 + x : (6,335 + 0,68x))))))$	$\Delta < 2 \cdot 10^{-4}$ при $1 < 1 + x < 10$

Полученные с помощью коррекции расчетные формулы некоторых элементарных функций, отличающиеся различной точностью, но требующие выполнения только арифметических операций, приведены в табл. 3 (в форме, удобной для вычислений на арифметических микрокалькуляторах).

В качестве аппроксимирующих выражений можно использовать не только степенные многочлены, а и другие функции, операторы вычисления которых входят в язык применяемого микрокалькулятора. Например, при наличии оператора извлечения квадратного корня для вычисления функции $\operatorname{arctg} x$ удобно использовать формулу

$$\alpha^\circ = \operatorname{arctg} x \approx 90x / \sqrt{a + x^2 + \sqrt{b + cx^2}},$$

которая при $a = 1,21163$; $b = 1,5762$ и $c = 1,6223$ обеспечивает погрешность $|\Delta| \leq 0,0006^\circ$ при любом значении аргумента. Для большинства практических задач удобно использовать ее упрощенный вариант

$$\alpha^\circ = \operatorname{arctg} x \approx 90x / \sqrt{1,2 + x^2 + \sqrt{1,62(1 + x^2)}}$$

с предельной погрешностью менее $0,03^\circ$.

Эти формулы представляют особый интерес для пользователей микрокалькулятора типа "Электроника Б3-21", словарь которого не содержит операторов вычисления обратных тригонометрических функций. В частности, последняя из формул реализуется программой, содержащей всего 23 оператора:

Программа вычисления функции $\operatorname{arctg} x$

P2 x^2 1 + 1 , 6 2 X $\sqrt{}$ + 5
1/x + $\sqrt{9}$ 0 XY \div ↑ F2 X C/P

Результат выполнения программы высвечивается в градусах. Эту программу можно использовать как самостоятельно, так и в качестве фрагмента более сложных программ.

2.2. В поисках корня

В игровых ситуациях очень часто возникает задача, когда необходимо за несколько попыток отгадать что-то. Покажем решение подобной задачи на следующем примере. Вы, читатель, со своими друзьями пошли в лес. Один из них предложил вам угадать за девять попыток число шагов до дерева, растущего у тропинки (число шагов было предварительно определено и составляло 511). На каждое называемое вами число он отвечал только "больше!", "меньше!" и "угадал".

Вы называете число 256, соответствующее середине длины тропинки. По ответу узнаете, в какой из ее половин растет это дерево и снова называете число шагов, при котором эта половина делится пополам. По ответу определяете, в какой из четвертей находится дерево. В конце концов даете правильный ответ менее даже чем за 9 попыток.

Советуем Вам сыграть несколько раз в подобную игру, чтобы на практике познакомиться с некоторыми ее тонкостями. А если Вы пользуетесь микрокалькулятором "Электроника Б3-21", то можно принять и его в качестве полноправного партнера. Для этого лишь введите в программную память следующую программу:

Программа игры в "больше-меньше"

5	1	2	P3	P2	↑	C/P	$x \neq 0$	P4	$x < 0$	RX	F3
+	P2	F3	2	÷	P3	↑	F2	XU	-	BП	F↑
F3	X	x^2	sin	x^2	1	0	2	2	X	1	+
1	BП	7	XU	-	-	P2	C _x	C/P	↑	F2	-
$x \neq 0$	PC _x	$x < 0$	P+	1	/-	BП	PC _x	1	BП	PC _x	

Теперь задумайте число и нажмите клавиши B/O и C/P. На индикаторе микрокалькулятора высветился его вопрос в виде числа. Если оно больше задуманного Вами, нажмите клавишу C/P и ожидайте следующего вопроса. Ес-

ли оно меньше задуманного, то сообщите об этом микрокалькулятору, нажав клавишу $/-$ изменения знака, после чего нажмите клавишу С/П. Если микрокалькулятор "угадал" задуманное число, сотрите содержимое индикаторного регистра, нажав клавишу C_x или 0 и нажмите клавишу С/П. В этом случае микрокалькулятор сам "задумает" число от 1 до 1023, которое Вы должны угадать.

Задавать вопросы микрокалькулятору следует вводом (набором числа-вопроса в регистр X). После этого, нажав клавишу С/П, Вы получите ответ микрокалькулятора в виде высвечиваемого им числа: -1, если "задуманное" им число больше числа-вопроса; 1, если меньше; 0, если Вы угадали "задуманное" число. Сравнив количества неудачных попыток, угадать число каждым партнером, Вы присуждаете победу себе (если Вам удалось быстрее угадать число, "задуманное" микрокалькулятором) или... повторяете партию — ведь обидно терпеть поражение от какой-то машинки.

Если Вы изучили стратегию игры в "больше—меньше", то можно перейти к более сложным проблемам. Но прежде постарайтесь ответить на вопрос: за сколько ходов можно гарантировать угадывание задуманного числа в описанной игре в самом "худшем" случае и почему? Дело в том, что подобным угадыванием по критериям типа "больше—меньше" приходится заниматься огромному числу людей, причем не ради развлечения, а с гораздо более серьезными намерениями.

Представьте себе, например, геологов, пробивающих шурфы в верхнем почвенном слое, чтобы взять пробы коренных пород в поисках верхней границы слоя нужного геологического возраста. Опытный геолог по этим пробам легко определяет, "старше" или "моложе" пробы нужной ему породы и, следовательно, получает указание, в каком направлении вести дальнейшие поиски.

Не раз Вам, вероятно, приходилось видеть рыбаков, щестами прощупывающих дно озера в поисках места с под-

ходящей глубиной. Многие читатели встречались с геодезистами, которые по измерениям высоты местности в нескольких точках определяют координаты мест с заданной высотой над уровнем моря. Что общего в действиях этих людей? Все они по пробам, дающим информацию типа "больше—меньше", стараются за минимальное число попыток определить координаты нужных им мест с определенными свойствами.

Но наибольшему числу людей приходится использовать подобные поисковые методы при решении математических уравнений или, другими словами, при поисках таких значений аргумента x , называемых *нулями* функции $y(x)$ или *корнями* уравнения $y(x) = 0$, при которых эта функция обращается в нуль.

Как ни покажется странным неискушенному в математике читателю, но почти все уравнения, встречающиеся в практических задачах, приходится решать способами, подобными используемым в игре "больше—меньше". Точные формулы, по которым значение корней может быть "вычислено", а не "угадано", существуют для весьма небольшой группы уравнений — к ней относится, например, и квадратное алгебраическое уравнение, хорошо известное школьникам. Но вычислительная практика свидетельствует, что и в этих случаях (например, при решении кубических уравнений) метод "угадывания" может оказаться предпочтительнее.

В общем случае зависимость физической величины y (например, высоты местности или возраста геологических пород) от координаты x , как и зависимость значений функции $y(x)$ от аргумента, можно отобразить графиком, на котором текущие значения y определяются относительно некоторого заданного (нулевого) уровня. Тогда точки пересечения этого графика с линией нулевого уровня (рис. 13) и будут искомыми корнями уравнения $y(x) = 0$.

На практике для облегчения любого поиска вначале

пытаются грубо оценить область, в которой находится искомый объект, а затем последовательно ее сократить (как в детском варианте игры в "больше–меньше", называемом "горячо–холодно") до решения задачи. Аналогично отыскивают и корни уравнений методами последовательных приближений.

При таком поиске по буквенной модели функции вначале пытаются оценить ее "поведение" и области существования нулей. Для этого, в частности, определяют знаки функций на границах интервала аргумента x , заданных условиями задачи (в частности, $x = -\infty$ и $x = \infty$), а также при тех значениях аргумента, для которых легко вычислить функцию (например, $x = 0$). Если на границах любого интервала $[x_a; x_b]$ значения функции противоположны по знаку ($y_a y_b < 0$), то в этом интервале функция обязательно принимает нулевое значение один или, в общем случае, нечетное число раз (сравните интервалы $[x_0; x_2]$ и $[x_0; x_4]$ на рис. 13). В противном случае ($y_a y_b > 0$) вопрос остается открытым – в интервале может содержаться либо четное число корней, либо ни одного корня (сравните интервалы $[x_0, x_1]$ и $[x_0, x_3]$ на рис. 13).

После грубой оценки интервала существования корней в нем выделяют более узкие интервалы, содержащие по

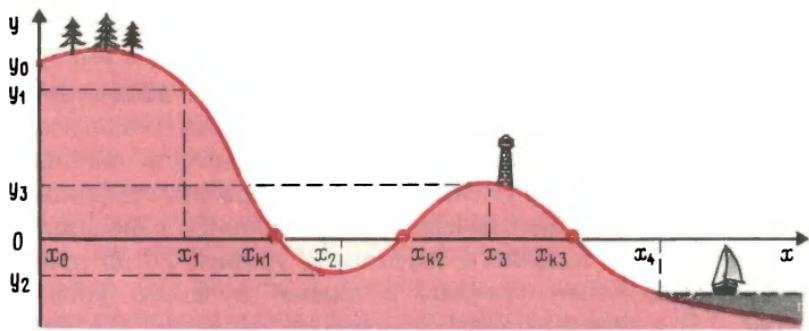


Рис. 13. Разрез местности по вертикали

одному корню (этап отделения корней). Для этого обычно вычисляют значение функции $y(x_p)$ для ряда значений аргумента x_p , начиная от края исходного интервала и сравнивая знаки двух очередных значений функции — если $y_p y_{p-1} < 0$, то корень находится между x_p и x_{p-1} (рис. 14, а).

Остается сократить ширину интервала, содержащего корень, так, чтобы после i -го сокращения она была меньше допустимой погрешности ϵ определения корня или $|\Delta x_i| = |x_i - x_{i-1}| \leq \epsilon$. С этой целью можно с успехом воспользоваться опытом игры в "больше—меньше" с половинным (или близким к нему) делением интервала. После каждого такого деления корень будет находиться в той части интервала, на границах которой значения функции противоположны по знаку (рис. 14, б).

Для иллюстрации займемся поиском корня алгебраического уравнения $0,000\ 0003z^3 + 0,0027z^2 + 0,99z + 195 = 0$. Многочлен со столь сильно отличающимися коэффициентами не вызовет у математиков положительных эмоций, так как его "неопрятность" свидетельствует, как правило, о неудачном выборе единиц измерения при постановке задачи. Возможно, например, что в приведенном уравнении переменная z означает длину, выраженную

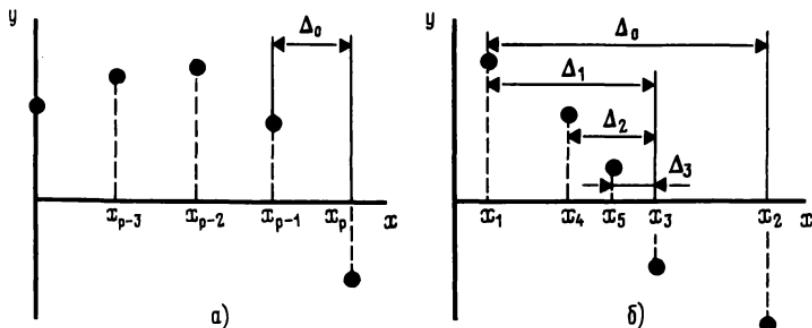


Рис. 14. Выделение интервала нахождения корня

ную в сантиметрах. Если перейти к измерению длины в метрах, обозначив ее как x , то она будет связана со "старой" длиной отношением $z = 100x$ и после подстановки его наше уравнение примет более изящный вид:

$$0,000\,0003 \cdot 1000000x^3 + 0,0027 \cdot 10000x^2 + 0,99 \cdot 100x + 195 = 3x^3 + 27x^2 + 99x + 195 = 0.$$

Можно еще больше упростить это уравнение, нормировав его делением на коэффициент при старшей степени аргумента:

$$y(x) \equiv x^3 + 9x^2 + 33x + 65 = 0,$$

так как при такой нормировке значения корней не изменяются.

Попытаемся грубо оценить область существования корней. Так как для нашего многочлена $y(-\infty) < 0$, $y(0) = 65 > 0$ и $y(\infty) > 0$, то, по крайней мере, один корень находится в интервале $]-\infty; 0[$. Этот интервал можно сузить, воспользовавшись теоремой, согласно которой модули нулей нормированных алгебраических многочленов не превышают более чем на единицу модуль наибольшего коэффициента a_m или $|x_k| \leq 1 + |a_m|$. Следовательно, корни или корень нашего уравнения лежат в интервале $]-66; 0[$.

Вычисление многочлена на микрокалькуляторе удобно выполнять по известной нам формуле, принимающей в данном случае вид

$$y(x) = ((x + 9)x + 33)x + 65.$$

Если Ваш микрокалькулятор имеет регистр памяти, то вычисления можно упростить, записав значение x в этот регистр. Например, на входном языке микрокалькулятора "Электроника Б3-18" вычисления удобно выполнять по программе

С x ЗАП + 9 X ИП + 3 3 X ИП + 65 =

Вычисления на микрокалькуляторе с многорегистровым операционным устройством (например, "Электроника Б3-19") упрощаются еще больше даже без обращения к памяти:

$$|C|x| \uparrow |9| + |X|3|3| + |X|6|5| +$$

Приняв на этапе отделения корней $x_0 = 0$ и постоянное приращение $\Delta x_0 = -2$, находим

$$y_1(-2) = ((-2+9)(-2)+33)(-2)+65 = 27; y_1 y_0 > 0;$$

$$y_2(-4) = ((-4+9)(-4)+33)(-4)+65 = 13; y_2 y_1 > 0;$$

$$y_3(-6) = ((-6+9)(-6)+33)(-6)+65 = -25; y_3 y_2 < 0.$$

Следовательно, искомый корень находится в интервале $]-6; -4[$. Поделив его пополам, вычисляем $y_4(-5) = ((-5+9)(-5)+33)(-5)+65 = 0$. Корень найден! Нам, конечно, повезло, но последовательно разделяя пополам очередной интервал, мы обязательно нашли бы корень с заданной точностью.

Попробуем оценить предельное число итераций для отыскания корня с абсолютной погрешностью Δ методом половинного деления при ширине исходного интервала Δx . При каждом делении интервал уменьшается вдвое и после k делений уменьшится в 2^k раз. Следовательно, число делений, требуемое для достижения допустимой погрешности, определяется неравенством $\Delta x / 2^k \leq \Delta$, откуда $k \geq (\ln x - \ln \Delta) / \ln 2$. Например, для вычисления корня решенного нами уравнения с погрешностью $\Delta \leq 10^{-8}$ при ширине исходного интервала $\Delta x = 2$ в худшем случае требуется $k = (\ln 2 + 8 \ln 10) / \ln 2 \approx 28$ итераций. При игре в "больше—меньше" с начальной шириной интервала $\Delta x = 1024$ и $\Delta = 1$ (так как учитываются только целые числа) требуется не более $k = \ln 1024 / \ln 2 = 10$ делений исходного интервала.

Аналогично определяется требуемое число делений пополам при допустимой относительной погрешности корня

$\delta \leq |\Delta x_k / x_k|$, но в этом случае необходимо знать величину корня. Для грубой оценки эту величину можно заменить значением аргумента в середине интервала. Таким образом, трудоемкость решения уравнений рассмотренным методом зависит от ширины начального интервала и требуемой точности вычисления корня.

Пользователь микрокалькулятора "Электроника Б3-21" может воспользоваться следующей программой, автоматически выполняющей сокращение исходного интервала нахождения корня до ширины, определяемой заданной точностью вычисления корня.

Программа вычисления корня уравнения $y(x) = 0$

```
F7` 2 ÷ P7 ↑ F8 + P8 ÷ x2 ↑ F6  
- x<0 X F8 C/P F8 ... ↑ F5 XY P5 X  
x<0 P0 F7 /-/ P7 B/O
```

Перед вычислениями в эту программу вместо многоточия записывают операторы вычисления левой части уравнения $y(x) = 0$ при значении аргумента x , хранящемся в регистре 8. Для записи коэффициентов функции $y(x)$ в качестве буферных могут быть использованы регистры 3, 4, 2 и стек памяти. После ввода программы в память заносят исходные данные: — начальное значение на нижней границе x_n в регистр 8, разность $x_b - x_n$ в регистр 7, квадрат допустимой относительной погрешности δ^2 в регистр 6, а число, совпадающее по знаку со значением $y(x_n)$ на нижней границе исходного интервала, в регистр 5. Вычисленное программой значение корня выводится на индикатор и хранится в регистре 8. При дополнительном нажатии клавиши С/П в регистр 5 заносится значение невязки $y(x_k)$.

Приведенную программу удобно использовать, если искомый корень не равен нулю, так как в противном случае возникнет переполнение. Поэтому в тех случаях,

когда начальный интервал охватывает нулевое значение аргумента, целесообразно заменить оператор \div по адресу 23 оператором F7, а в регистр 6 перед пуском программы заносить допустимую абсолютную погрешность Δx_k , возведенную в квадрат. Вблизи корня значения функции $y(x)$ стремятся к нулю, но выбор критерия $y(x) = 0$ для прекращения поиска корня может привести не только к большой погрешности определения корня при медленном изменении функции вблизи него, но и к "зацикливанию" программы, когда вычисления не прекращаются в связи с невозможностью выполнения условия $y(x) = 0$ или $y(x) \leq \epsilon$ в связи с быстрым изменением функции вблизи корня (рис. 15). В последнем случае ширина интервала нахождения корня будет определяться машинным нулем A_{\min} , но вычисления не прекратятся.

Рассмотрим два примера практического применения рассматриваемой программы.

1. Составим программу автоматического решения уравнения $\operatorname{tg} x - x - |A| = 0$, ограничившись поиском корня, содержащегося в интервале $[0; \pi/2]$.

Используя для хранения постоянного коэффициента $|A|$ регистр 2, составим для вычисления левой части реш-

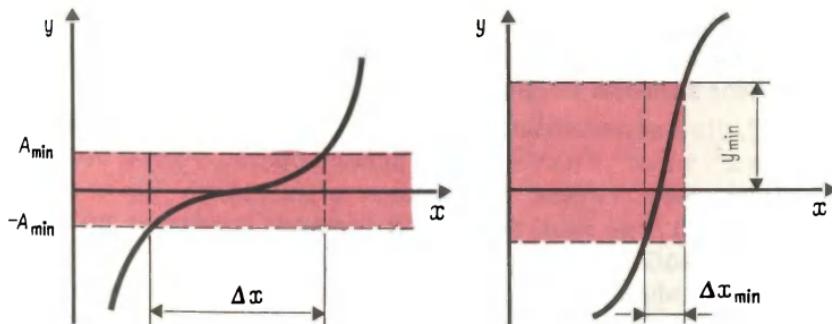


Рис. 15. Зависимость определения корня от поведения функции

емого уравнения фрагмент

F8 $e^{jx} \div \uparrow$ F8 $- \uparrow$ F2 $-$

и запишем его (кроме уже имеющегося в программе первого оператора F8) в программу на место многоточия. Если уравнение предполагается решать неоднократно при различных значениях постоянной $|A|$, то ввод исходных данных $x_h = 0$, $x_b - x_h = \pi/2, -\pi/2$ (отрицателен знак невязки уравнения на левой границе интервала, содержащего корень) осуществляется в регистры 8, 7 и 5 соответственно, а засылку значения $|A|$ из регистра X в регистр 2 целесообразно автоматизировать, составив следующую программу.

Программа решения уравнения $\operatorname{tg} x = x + |A|$

P2	Cx	P8	2	\div	P7	$/-$	P5	F7	2	\div
P7	\uparrow	F8	P8	\div	x^2	\uparrow	F6	$-$	$x < 0$	P,
F8	C/P	F8	e^{jx}	\div	\uparrow	F8	$-$	\uparrow	F2	$-$
F5	XY	P5	\times	$x < 0$	1	F7	$/-$	P7	BП	1

После ввода этой программы в программную память заносят в регистр 6 квадрат допустимой погрешности корня ϵ^2 , в регистр X значение постоянной $|A|$ и нажимают клавиши В/О и С/П. При $|A| = 10$ $\epsilon^2 = 10^{-12}$ примерно через три минуты на индикаторе вы светится вычисленное значение корня $x = 1,483937$.

2. Для автоматизации решения алгебраического уравнения $x^3 + a_2x^2 + a_1x + a_0 = 0$ целесообразно учесть, что вещественный корень расположен в интервале $[0, \pm(1 + |a_{\max}|)]$, где знак ненулевой границы совпадает по знаку со свободным членом a_0 , а $|a_{\max}|$ – наибольший модуль коэффициентов. Размещая исходные данные $a_2 = -P2$, $a_1 = P3$, $a_0 = P4$, $\pm(1 + |a_{\max}|) = P7$, квадрат допустимой погрешности $\epsilon^2 = P6$, можно составить следующую программу.

Программа вычисления корня уравнения $x^3 + a_2x^2 + a_1x + a_0 = 0$

```
Cx P8 F4 P5 x>0 XY /-/ P5 F7 /-/ P8 F7
  2 ÷ P7 ↑ F8 + P8 ÷ x2 ↑ F6 -
x<0 F, F8 C/P F8 ↑ F2 + X ↑ F3 +
↑ F8 X ↑ F4 + ↑ F5 XY P5 X x<0
XY F7 /-/ P7 БП XY
```

По этой программе при $76 = P7, 1 \cdot 10^{-16} = P6$ корень $x = -5$ уравнения $x^3 + 9x^2 + 35x + 75 = 0$ вычисляется примерно за три минуты.

Однако можно значительно быстрее получить значение корня, если в интервале $[x_a; x_b]$ провести интерполяирующую прямую (рис. 16, а). Ее пересечение с осью аргумента даст приближенное значение корня x_1 . Взяв его в качестве новой границы сокращаемого интервала и повторяя интерполяцию, можно с любой точностью вычислить корень

$$x_i = (y_0 x_{i-1} - y_{i-1} x_0) / (y_0 - y_{i-1}); i = 1, 2, \dots$$

В качестве неподвижной границы x_0 следует выбирать то из значений x_a или x_b , для которого знак функции противоположен знаку $y(x_1)$.

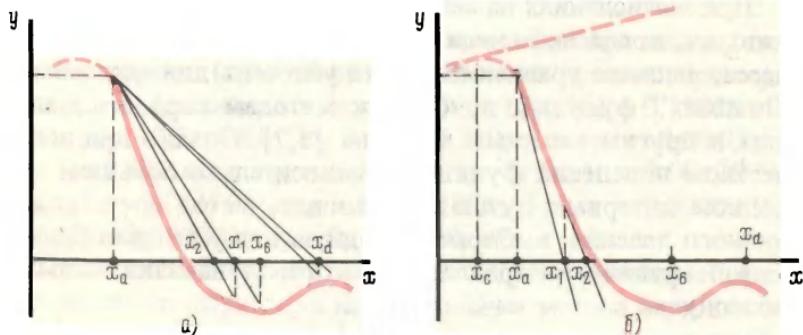


Рис. 16. Зависимость сходимости методов хорд и касательных от ширины исходного интервала

Но можно найти корень еще быстрее, если на границе x_a аппроксимировать функцию линейным многочленом Тэйлора $y(x) \approx y(x_a) + y'(x_a)(x - x_a)$, описанным вами в предыдущем разделе. Приняв $x = x_a - y(x_a)/y'(x_a)$ при $y(x_1) = 0$ в качестве новой границы интервала, легко найдем последующие приближения

$$x_i = x_{i-1} - y(x_{i-1})/y'(x_{i-1}); i = 1, 2, \dots$$

Известно, что производная численно равна наклону касательной к графику функции (рис. 16, б) и даже на глаз видно, как быстро сходится приближение.

Собравшись с мыслями, мы процитировали слова известного математика: "Чем тоньше метод и чем лучше он кажется, тем хуже он может повести себя присложнениях с функцией"¹. Изложенные методы известны под названием методов хорд и касательных и, действительно, быстро сходятся в малых интервалах аргумента, где функция близка к линейной. Однако при больших интервалах (например, интервал $[x_c, x_d]$ на рис. 16, б) эти методы могут сходиться медленно или даже расходиться, тогда как метод деления интервала, не зависящий от вида функции, достаточно надежен и обеспечивает сходимость даже при автоматических вычислениях.

При вычислениях на непрограммируемых микрокалькуляторах, когда возможен контроль вычислительного процесса, решение уравнений можно ускорить для достаточно "гладких" функций, прибегая к методам хорд, касательных и другим сложным методам [3,7]. Однако при неизвестном поведении функции в относительно большом исходном интервале лучше использовать метод почти половинного деления, выбирая очередную точку деления ближе к той границе интервала, для которой невязка меньше по модулю.

¹ Р. В. Хеминг. Численные методы. – М.: Наука, 1968.

Для сравнения на микрокалькуляторе с оператором e^x решим различными методами уравнение $e^x - 2 = 0$ при допустимой погрешности корня $\delta \leq 10^{-4}$. Выбрав начальный интервал $[0; 1]$ по невязкам $y(0) = e^0 - 2 = -1$ и $y(1) = 0,718281$, найдем методом почти половинного деления: $y_1(0,5) = -0,351279$; $y_2(0,75) = 0,1169995$; $y_3(0,625) = -0,1317544$; $y_4(0,7) = 0,0137523$; $y_5(0,68) = -0,0261228$; $y_6(0,69) = -0,0062849$; $y_7(0,695) = 0,0037086$; $y_8(0,693) = -0,0002947$; $y_9(0,6935) = 0,0002947$; $y_{10}(0,6932) = 0,0001051$; $y_{11}(0,6931) = -0,0000949$; $y_{12}(0,69315) = -0,0000051$. Погрешность последнего приближения $|\Delta x_{12}/x_{12}| = 0,73 \cdot 10^{-4} < 10^{-4}$ и, следовательно, корень вычислен с требуемой точностью.

В том же исходном интервале методом хорд находим первое приближение $x_1 = (-1)(-0,718281)/(-1-0,718281) = 0,5819679$. Так как $y(x_1) = -0,2104276$ по знаку противоположно значению $y(1) = 0,718281$, то выбираем в качестве неподвижной границы $x = 1$ и вычисляем: $x_2 = 0,6755929$; $y(x_2) = -0,0326398$; $x_3 = 0,6896936$, $y(x_3) = -0,0068958$; $x_4 = 0,6926042$, $y(x_4) = -0,0010861$; $x_5 = 0,6930682$, $y(x_5) = -0,0001584$; $x_6 = 0,6931358$, $y(x_6) = -0,0000232$; $x_7 = 0,6931457$, $y(x_7) = -0,0000034$ с относительной погрешностью $|(x_7 - x_6)/x_7| = 0,14 \cdot 10^{-4} < 10^{-4}$.

При использовании метода касательных определяем, что производная $y'(x) = e^x$ на границе $x = 0$ исходного интервала равна единице. Тогда $x_1 = 1/1 = 1$, $y(x_1) = 0,7182814$, $y'(x_1) = 2,718281$; $x_2 = 0,735759$, $y(x_2) = 0,087065$, $y'(x_2) = 2,087065$; $x_3 = 0,6940425$, $y(x_3) = 0,0017909$, $y'(x_3) = 2,001790$; $x_4 = 0,6931478$, $y(x_4) = 0,0000008$, $y'(x_5) = 2,000000$; $x_5 = 0,6931474$, $y(x_5) = 0$.

Приведенный пример наглядно подтверждает уменьшение числа итераций при использовании метода хорд и особенно касательных. Однако и в этом случае метод почти половинного деления отличается простотой вычисления.

2.3. Сколько было винограда, или как решить систему уравнений?

Читатель знает, что решением или корнями системы уравнений называют такие значения неизвестных, при которых одновременно удовлетворяются все уравнения и их невязки (разности правой и левой частей) равны нулю.

В качестве примера решения системы уравнений рассмотрим следующую задачу:

покупатель давал два рубля за яблоки и три за виноград, а за все фрукты 12 рублей, но продавец хотел получить вдвое больше и просил за яблоки четыре рубля и за виноград шесть. Сколько же было килограммов винограда у продавца? Предложения покупателя и продавца можно описать уравнениями

$$2x_1 + 3x_2 = 12; \quad 4x_1 + 6x_2 = 24,$$

обозначив символами x_1 и x_2 соответственно неизвестные значения винограда и яблок в килограммах.

Все попытки решить составленную систему линейных уравнений не приводят быстро к успеху – для любого, даже дробного значения одной переменной.

Если продавец уменьшит стоимость винограда и яблок на рубль, а покупатель заплатит за все фрукты 19 рублей, то задача может быть описана следующими уравнениями:

$$2x_1 + 3x_2 = 12; \quad 3x_1 + 5x_2 = 19.$$

В этом случае быстро находится решение: 3 кг яблок и 2 кг винограда.

Попытаемся разобраться в затруднениях, возникающих при решении этой задачи. Уравнение называют линейным, если неизвестные связаны линейными зависимостями. Такое уравнение с двумя неизвестными легко отобразить на плоскости с прямоугольной системой координат x_1 и x_2 прямой линией. Каждой точке этой прямой соответствуют ее координаты, являющиеся решением уравнения. Следова-

тельно, решением системы из двух уравнений будут координаты точки, через которые проходят прямые, отображающие уравнения. Но две прямые могут быть проведены на плоскости тремя способами:

прямые совпадают и координаты любой их точки являются решением — система имеет бесконечное число решений, а ее уравнения отличаются лишь постоянным множителем; с подобной ситуацией мы сталкиваемся, решая первые из составленных уравнений (рис. 17, а);

прямые, отображающие уравнения, параллельны. Система не имеет решения, а их левые (содержащие неизвестные) части имеют общий множитель; с этим случаем встретимся, изменив свободный член в одном из уравнений (рис. 17, б);

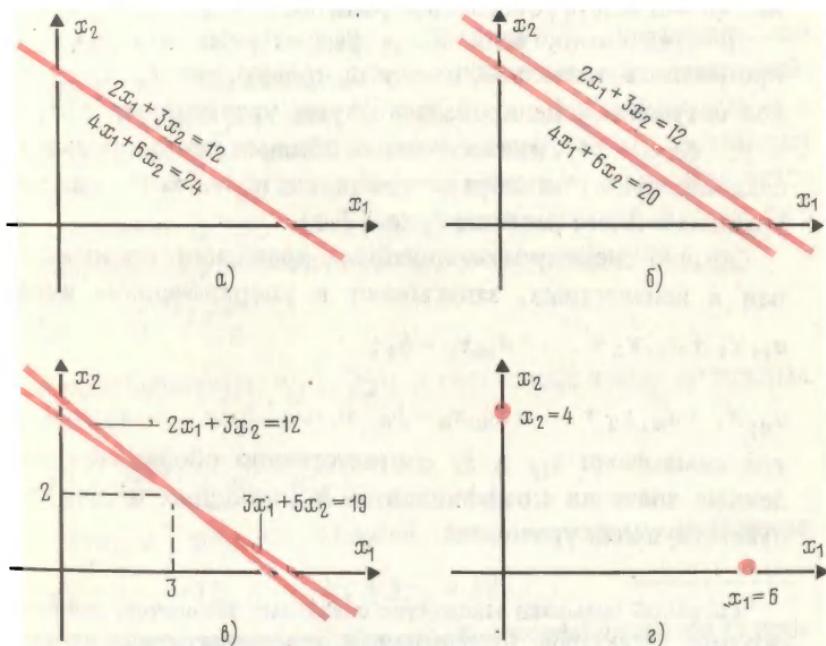


Рис. 17. Представление линейных уравнений на плоскости

прямые пересекаются в одной точке (рис. 17, в), соответствующей единственному решению системы уравнений.

Уравнения, левые части которых отличаются лишь постоянным множителем, математики называют линейно- зависимыми. Система линейных уравнений имеет единственное решение при линейной независимости уравнений. Это условие достаточное, а необходимым условием единственности решения является связность уравнений. Любое множество объектов – это система¹ при наличии признака или свойства, связывающего эти объекты. Два уравнения связаны, если они содержат хотя бы по одному не нулевому члену с одной и той же неизвестной. В противном случае эти уравнения не образуют систему и каждое из них может иметь собственное решение.

Действительно, если бы в рассматриваемом примере продавались только яблоки или только виноград, то такая ситуация моделировалась двумя уравнениями ($2x_1 = 12$; $6x_2 = 24$), не связанными общими неизвестными и, следовательно, не образующими систему, хотя каждое уравнение имеет решение (рис. 17, г).

Обычно систему из n линейных уравнений, пронумеровав n неизвестных, записывают в упорядоченном виде:

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1;$$

...

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n,$$

где символами a_{ij} и b_i соответственно обозначены численные значения коэффициентов и свободных членов соответствующих уравнений.

¹ Системой называют множество связанных элементов, характеризуемое структурой (определяющей отношения между элементами) и параметрами (определяющими свойства элементов).

Совокупность точек, координаты которых удовлетворяют линейному уравнению с тремя неизвестными, образуют в трехмерном пространстве плоскость. При $n > 3$ совокупность точек абстрактного n -мерного пространства нельзя представить геометрическим образом, но условия единственности решения системы линейных уравнений справедливы для любого n . В этом случае коэффициенты a_{ij} могут иметь любые значения, включая и нулевые, при которых выполняются условия связности и линейной независимости уравнений.

Среди многочисленных способов решения систем линейных уравнений для вычислений на микрокалькуляторах удобен метод Жордана–Гаусса. Он заключается в преобразовании за n шагов исходной системы уравнений к виду $x_i = b_i^{(n)}$ ($i = 1, 2, \dots, n$), где $b_i^{(n)}$ – численные значения преобразованных свободных членов уравнений.

На каждом p -м шаге ($p = 1, 2, \dots$) преобразования в очередном p -м столбце коэффициентов a_{ip} сохраняют только $a_{pp}^{(p)} = 1$, а остальные заменяют нулями. Для этого остальные ($j > p$) коэффициенты и свободный член p -го (ведущего) уравнения преобразуют по формуле

$$\alpha_{pj}^{(p)} = \alpha_{pj}^{(p-1)} / \alpha_{pp}^{p-1},$$

а коэффициенты a_{ij} ($j > p$) и свободные члены остальных уравнений – по формуле

$$a_{ij}^{(p)} = a_{ij}^{(p-1)} - \alpha_{ip}^{p-1} \alpha_{pj}^p.$$

Решим для наглядности этим методом уравнения
 $2x_1 + 3x_2 = 12;$ $3x_1 + 5x_2 = 19.$

После исключения первого столбца коэффициентов остальные элементы первого (ведущего) уравнения $\alpha_{12}^1 a_{12} / a_{11} = 6$ Зак. 953

$= 3/2$; $b_1^1 = b_1/a_{11} = 12/2 = 6$, а элементы второго $\alpha_{22}^1 = a_{22} - a_{21}\alpha_{12}^1 = 5 - 3 \cdot 3/2 = 1/2$; $b_2^1 = b_2 - a_{21}b_1^1 = 19 - 3 \cdot 6 = 18$. Следовательно,
 $x_1 + 1,5x_2 = 6$; $0 + 0,5x_2 = 1$.

После исключения второго столбца коэффициентов элементы ведущего (второго) уравнения $b_2^{(2)} = b_2^{(1)} / \alpha_{22}^1 = 1/0,5 = 2$, а первого $b_1^{(2)} = b_1^{(1)} - \alpha_{12}^{(1)} b_2^{(2)} = 6 - 1,5 \cdot 2 = 3$, или $x_1 + 0 = 2$; $0 + x_2 = 3$.

При решении системы из n линейных уравнений рассматриваемым методом обычно составляют вычислительный бланк из $n + 1$ частей, каждая из которых содержит n строк и $n + 2$ столбца. В первые n столбцов начальной (нулевой) части бланка записывают коэффициенты заданной системы уравнений, в следующий ($n + 1$)-й столбец — свободные члены, а в ($n + 2$)-й контрольный столбец — суммы остальных элементов строк. Примером может служить запись коэффициентов и свободных членов системы

$$\begin{aligned} 10x_1 - x_2 + 2x_3 - 3x_4 &= 0; \\ x_1 + 10x_2 - x_3 + 2x_4 &= 5; \\ 2x_1 + 3x_2 + 20x_3 - x_4 &= -10; \\ 3x_1 + 2x_2 + x_3 + 20x_4 &= 15 \end{aligned}$$

в начальную часть бланка, приведенного в табл. 4.

При заполнении каждой p -й части бланка исключают p -й столбец коэффициентов, записывая $\alpha_{pp}^{(p)} = 1$ и $\alpha_{ip}^{(p)} = 0$, и по приведенным выше формулам вычисляют и записывают в бланк остальные элементы. Вычисленные по этим формулам элементы контрольного столбца должны быть равны суммам остальных элементов строк — нарушение этого правила свидетельствует об ошибке в вычислениях. Значения искомых переменных равны преобразованным свободным членам в p -й части бланка. Число элементов

**Вычислительный бланк для решения системы линейных
уравнений методом Жордана–Гаусса**

<i>p</i>	<i>i</i>	<i>x₁</i>	<i>x₂</i>	<i>x₃</i>	<i>x₄</i>	<i>b</i>	Σ
0	1	10	-1	2	-3	0	8
	2	1	10	-1	2	5	17
	3	2	3	20	-1	-10	14
	4	3	2	1	20	15	41
1	1	1	-0,1	0,2	-0,3	0	0,8
	2	0	10,1	-1,2	2,3	5	16,2
	3	0	3,2	19,6	-0,4	-10	12,4
	4	0	2,3	0,4	20,9	15	38,6
2	1	0	0	0,1881188	-0,2772277	0,04950495	0,960396
	2	0	1	-0,1188118	0,2277227	0,49504950	1,60396
	3	0	0	19,98019	-1,128712	-11,58415	7,267327
	4	0	0	0,6732673	20,37623	13,86138	34,91089
3	1	1	0	0	-0,2666005	0,1585728	0,8919721
	2	0	1	0	0,2210108	0,4261645	1,647175
	3	0	0	1	-0,05649155	-0,5797817	0,3637266
	4	0	0	0	20,41426	14,25172	34,666
4	1	1	0	0	0	0,3446934	1,344693
	2	0	1	0	0	0,2718711	1,27187
	3	0	0	1	0	-0,5403435	0,4596563
	4	0	0	0	1	0,6981257	1,698126

строк уменьшается в следующей части бланка, тогда как число строк остается неизменным. Поэтому целесообразно заполнять каждую часть бланка по столбцам, предварительно вычислив элемент ведущей строки.

ты этого столбца вводят против часовой стрелки в стек памяти, "доворачивая" его при $n < 7$ так, чтобы введенное первым число оказалось в регистре $C1$ (см. рис. 9). После этого в регистр X вводят первый элемент следующего столбца предыдущей части и нажимают клавиши В/О и С/П, что приводит к высвечиванию нуля.

Дальнейшие вычисления элементов p -й части сводятся к вводу в регистр X по столбцам остальных элементов, нажатию клавиш С/П и регистрации в бланке высвечиваемых результатов на "месте" предыдущего элемента предыдущей части бланка. После ввода последнего элемента $(p+2)$ -го столбца результат вычислений записывают как $\alpha_{n-1,n+2}$, после чего нажимают клавишу С/П и регистрируют высвечиваемое значение $\alpha_{n,n+2}$. Например, для вычисления первой части табл. 5 следует ввести операторы (в скобках указаны регистрируемые результаты вычислений): 4 P8 2 0 P2 1 \leftarrow 2 \leftarrow 1 $/-$ \leftarrow \leftarrow \leftarrow 1 $/-$ В/О С/П (0) 2 0 С/П (20,05) 3 $/-$ С/П (-2,9) 2 С/П (1,95) 2 С/П (-0,05) 3 С/П (2,9) . . . 1 7 С/П (17,7) С/П (0,7).

В процессе вычислений по этой программе строки циклически переставляются, но в n -й части бланка исходный порядок строк восстанавливается.

Точность решения системы уравнений обычно оценивают по невязкам уравнений после подстановки вычисленных корней. Для уточнения результатов невязки записывают в дополнительный столбец бланка и над ними выполняют те же операции, что и над остальными элементами — тогда в последней части бланка будут вычислены поправки к соответствующим корням. Однако такой способ уточнения решения не всегда применим. Для примера попытаемся решить систему уравнений $0,09954x_1 - 0,30012x_2 = 0,09504$; $0,05043x_2 - 0,15205x_2 = 0,04815$ на восьмиразрядном микрокалькуляторе с естественным представлением чисел. При подстановке x_1 из первого уравнения

во второе получим $x_2 = 1$, $x_1 = 3,9698613$. Если подставить x_2 из первого уравнения во второе, то получим $0 = 0,0000001$.

Погрешности округления уменьшаются при умножении рассматриваемых уравнений на 100 или использовании микрокалькулятора с показательной формой представления чисел. Но и тогда по первому способу получим $x_1 = 8,9949765$ и $x_2 = 2,666666$, а по второму $x_1 = 14,583333$ и $x_2 = 4,5201413$. Между тем, решив ту же систему уравнений при помощи карандаша и бумаги без округления результатов, получим $x_1 = 10$ и $x_2 = 3$.

Причина столь сильного влияния погрешностей округления становится очевидной, если изобразить решаемые уравнения прямыми линиями, как на рис. 17. Okажется, что прямые пересекаются под столь малым углом, что в широком интервале изменения переменных расстояние между прямыми меньше машинного нуля микрокалькулятора. Следовательно, хотя для таких систем уравнений условие единственности решения выполняется, оно нарушается при вычислениях с ограниченной разрядностью. Поэтому подобные системы математики называют плохо обусловленными.

Уравнение с n неизвестными формально соответствует системе из n совпадающих уравнений. Хотя такие системы, как уже известно, имеют бесконечное число решений, в практических задачах могут иметь смысл лишь некоторые из них, например целочисленные. Уравнения вида $a_1x_1 + \dots + a_nx_n = b$ с целочисленными коэффициентами, называемые неопределенными или диофантовыми, разрешимы в целых числах, если свободный член делится на наибольший общий делитель коэффициентов.

Особенности неопределенных уравнений позволят Вам удивить товарища вычислением его дня и месяца рождения по одному сообщенному им числу. Для этого попросите его умножить день рождения на 12, а номер месяца —

на 31 и сообщить Вам результат сложения этих чисел, а сами занесите этот результат в регистр X микрокалькулятора "Электроника Б3-21" с ранее введенной следующей программой.

Программа вычисления зашифрованных дат

P4	C _x	P5	F5	1	+	P5	3	1	X	↑	F4
XY	-	1	2	÷	P8	1	BП	7	XY	-	-
↑	F8	-	x=0	0	F5	1	0	÷	↑	F8	+
C/П БП РО											

После нажатия клавишей B/0 и C/П через несколько десятков секунд на индикаторе вы сможете прочесть дату рождения своего товарища. Например, по числу 378 микрокалькулятор вычислит 16,6, что соответствует дате 16 июня.

Выполняя программу, микрокалькулятор решает неопределенное уравнение $12x + 31y = A$, где A – исходное число, а x и y – неизвестные день и месяц рождения. Для этого программа вычисляет $x = (A - 31y)/12$, последовательно подставляя $y = 1, 2, \dots$ и проверяя, является ли результат вычислений x целым числом. Коэффициенты исходного уравнения подобраны так, что среди имеющихся смыслов решений, на которые наложены очевидные ограничения $0 < x \leq 31$ и $0 < y \leq 12$, обязательно встретится целочисленное и единственное в указанных пределах.

При использовании непрограммируемого микрокалькулятора с регистром памяти для решения подобной задачи можно вводить предложение

$A - 31 \div \text{ЗАП } 12 = (?)$ ИП – $31 \div \text{ЗАП } 12 = \dots$,
повторяя эту последовательность до получения целого числа x , после чего месяц рождения определяется по формуле $y = (A - 12x)/31$.

2.4. Немного статистики

В окружающем нас мире любое явление есть причина и следствие бесконечного множества других явлений. Но именно в силу этого бесконечного множества причин, влияющего на каждое явление, оно становится случайным и подчиняется вероятностным законам. Нажимая кнопку у двери знакомых, мы ожидаем, как безусловного следствия, звукового сигнала. Но он может и не прозвучать, так как по каким-то случайным причинам отказал механизм звонка или отсутствовало его питание. Следовательно, даже эта простейшая причинно-следственная связь (между нажатием кнопки и звонком) является случайной и реализуется лишь с определенной вероятностью, которую можно оценить статистическими методами.

Статистические оценки мы часто используем в повседневной жизни не только при взвешивании своих шансов на поступление в институт, но даже при длительном ожидании троллейбуса. Однако подобные оценки, как и само понятие вероятности, мы определяем чисто интуитивно, часто не подозревая, в какие строгие количественные формы облечает их теория вероятности и такой ее раздел, как математическая статистика. Между тем ознакомление даже с простейшими вероятностными и статистическими закономерностями может существенно облегчить оценку результатов наблюдений и принятие соответствующих решений.

В теории вероятности изучаются случайные события — любые явления, которые в результате воспроизводимого эксперимента могут произойти или не произойти. Предельными случаями случайных событий являются *достоверное* событие — такое, которое происходит непременно, и *невозможное* — не происходящее никогда. События, которые не могут произойти одновременно, называют *несовместимыми*.

С каждым из возможных событий x_i связывается

число $P(x_i)$, называемое вероятностью этого события и характеризующее среднее число (частоту) появления события x_i в результате многократного повторения эксперимента. Например, если вероятность события a равна $P(a) = 0,3$, то при N -кратном повторении оно в среднем должно произойти $0,3 \cdot N$ раз. Очевидно, что для $P(a)$ должно выполняться соотношение $0 \leq P(a) \leq 1$. Если исходом эксперимента обязательно является одно из несовместимых событий x_j и никакие другие события произойти не могут, говорят, что множество событий $\{x_j\}$ образуют *полную группу*. В этом случае должно выполняться соотношение $\sum P(x_j) = 1$, называемое условием нормировки и означающее, что появление какого-либо события из полной группы есть событие достоверное, вероятность которого, очевидно, должна равняться единице.

Во многих практических задачах результатом эксперимента со случайным исходом является некоторое число (например, показания термометра или цифры в спорто-лoto). Однако в тех случаях, когда результат опыта непосредственно не имеет числового выражения, каждый из его возможных исходов можно "пронумеровать", как это сделано, например, с гранями игральной кости. Поэтому далее под x_i будем понимать случайную величину, а частоту появления каждого из ее возможных значений будем обозначать как функцию $P(x_i)$, называемую законом распределения.

Если случайная величина принимает только конечное или счетное значение x_1, x_2, \dots, x_n , то ее называют дискретной. Наряду с дискретными часто приходится встречаться со случайными величинами, которые могут принимать любое значение в некотором интервале (включая и весь бесконечный интервал чисел). Такие случайные величины называют непрерывными. Поскольку в любом непрерывном интервале "вмещается" бесконечно много чисел, то каждому точному значению случайной величины прихо-

дится приписать равную нулю вероятность. Отличной от нуля в этом случае будет, например, вероятность того, что случайная величина окажется меньше заданного значения ξ . Эта вероятность характеризуется функцией $F(x) = P(x < \xi)$.

Функция $F(x)$, называемая интегральным законом распределения, монотонно возрастает при увеличении x и при $F(-\infty) = 0$, а при $F(\infty) = 1$. С помощью этой функции легко вычислить вероятность попадания случайной величины в заданный интервал $[x_1; x_2]$ как $P(x_1 \leq x < x_2) = F(x_2) - F(x_1)$. Однако в практике для этого чаще используют функцию плотности вероятности $p(x)$, связанную с функцией $F(x)$ соотношением $p(x) = dF(x)/dx$ и обладающую свойствами неотрицательности и условием нормировки $\int_{-\infty}^{\infty} p(x) dx = 1$. Для вероятности попадания случайной величины в интервал $[x_1; x_2]$ справедливо соотношение

$$P(x_1 \leq x < x_2) = \int_{x_1}^{x_2} p(x) dx.$$

Законы распределения случайных величин определяются их природой и, строго говоря, могут быть найдены только экспериментально. Однако в некоторых случаях условия симметрии позволяют приписать всем случайным величинам, образующим полную группу, равную вероятность. Например, разумно предположить, что при бросании игральной кости "выпадение" любой ее грани равновероятно. Тогда из условия нормировки каждому числу очков, написанному на ее гранях и образующих полную группу из шести дискретных значений, следует приписать вероятность $P(x_i) = 1/6$; $x \in \{1, 2 \dots 6\}$. Закон распределения этой дискретной случайной величины изображен на рис. 18, a.

Нарушение симметрии реальной кости приведет к изменению вероятностей выпадания различного числа очков (например, такому, как показано на рис. 18, б) и может быть обнаружено только экспериментально.

Однако определение вероятностных характеристик методом статистических испытаний часто требует очень большого (более миллиона) числа экспериментов и не всегда выполнимо практически. Однако во многих случаях, анализируя причинно-следственные связи рассматриваемого события, вероятность его появления удается связать с вероятностью других, более "простых" событий, вероятность которых либо известна, либо проще определяется экспериментально. Вот тут-то во всю свою мощь проявляется практическая ценность теории вероятности, позволяющая в этих случаях найти интересующий исследователя закон распределения. Более того, во многих практических задачах теоретический закон распределения бывает заранее известен и экспериментально приходится лишь определять его числовые характеристики. Например, равномерный закон распределения однозначно определяется его границами x_a и x_b , а гауссовский

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-a)^2/2\sigma^2}$$

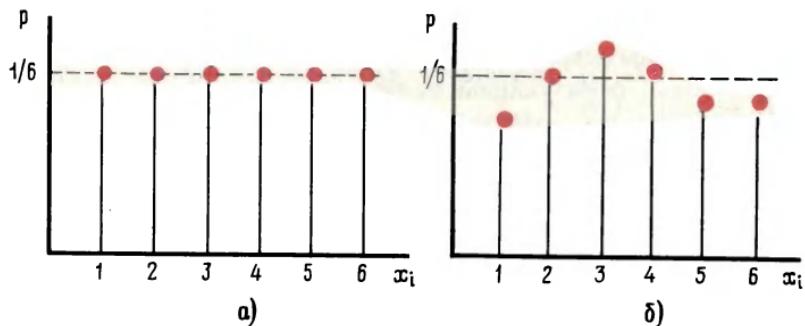


Рис. 18. Распределение вероятностей при бросании кубика

числовыми коэффициентами a и σ . Поэтому при статистической обработке результатов в первую очередь (и этого иногда бывает достаточно) определяют следующие характеристики случайной величины:

математическое ожидание или среднее значение, связанное с законом распределения дискретных величин соотношением $m_1 = \sum_{i=1}^{\infty} P(x_i)x_i$, а с функцией плотностей вероятности $m_1 = \int_{-\infty}^{\infty} xp(x) dx$;

дисперсия (равная квадрату наиболее вероятного среднеквадратического отклонения σ случайной величины от ее среднего значения), определяемая по законам распределения дискретной и непрерывной случайной величины соответственно выражениями

$$\sigma^2 = \sum_i (x_i - m_1)^2 P(x_i) \text{ и } \sigma^2 = \int_{-\infty}^{\infty} (x - m_1)^2 p(x) dx.$$

По результатам эксперимента можно получить оценки этих числовых характеристик:

$$\tilde{m}_1 = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{и} \quad \tilde{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \tilde{m}_1)^2$$

(в отличие от точных значений оценки обозначают волнистой линией сверху). Значения экспериментально найденных оценок оказываются тем точнее, чем больше число испытаний (наблюдений или измерений) N проведено для получения значений x_i случайной величины. При ограниченном числе испытаний возможную ошибку оценки среднего значения (равную среднеквадратическому отклонению, обусловленному случайностью используемой выборки данных) можно определить по оценке дисперсии как $\tilde{\sigma}_m = \sqrt{\tilde{\sigma}^2/N}$.

Часто возможную ошибку оценки определяют по доверительному интервалу, в котором с заданной вероятностью (с заданным "доверием") $P_{\text{дов}}$ находится истинное зна-

чение оцениваемой величины. Для вычисления границ доверительного интервала среднего используют формулу $\tilde{m}_1 - t\tilde{\sigma}/\sqrt{N} \leq m_1 \leq \tilde{m}_1 + t\tilde{\sigma}/\sqrt{N}$, а доверительный интервал для оценки среднеквадратического отклонения находят по соотношению $\tilde{\sigma}(1-q) \leq \sigma \leq \tilde{\sigma}(1+q)$. Приближенные значения коэффициентов t и q , зависящие от используемого числа испытаний N и доверительной вероятности $P_{\text{дов}}$ для относительно небольшого числа N , приведены в табл. 6.

Одновременное вычисление оценок \tilde{m}_1 и $\tilde{\sigma}^2$, обычно требуемое при решении практических задач, проще организовать при использовании микрокалькулятора с регистром памяти. Вместо вычисления оценки $\tilde{\sigma}^2$ по приведен-

Таблица 6

Зависимость коэффициентов t и q от числа N испытаний

N	$P_{\text{дов}}$					
	0,95		0,99		0,999	
	t	q	t	q	t	q
20	2,086	0,336	2,845	0,518	3,850	0,776
30	2,042	0,276	2,750	0,416	3,646	0,609
40	2,021	0,234	2,704	0,343	3,551	0,494
50	2,008	0,207	2,677	0,297	3,497	0,424
60	2,000	0,187	2,660	0,266	3,460	0,375
70	1,995	0,172	2,648	0,242	3,436	0,339
80	1,990	0,160	2,639	0,224	3,416	0,311
90	1,987	0,150	2,632	0,209	3,401	0,289
100	1,984	0,142	2,626	0,196	3,391	0,271
150		0,115		0,159		0,220
200		0,099		0,135		0,184
250		0,089		0,120		0,162
300		0,081		0,109		0,146
∞	1,960	0,0	2,576	0,0	3,291	0,0

ной выше формуле в этом случае целесообразно воспользоваться соотношением, справедливость которого легко докажет читатель:

$$\tilde{\sigma}^2 = \left(\sum_{j=1}^N x_j^2 - N\tilde{m}_1^2 \right) / (N-1).$$

Тогда, например, на микрокалькуляторе "Электроника Б3-18" можно одновременно вычислять суммы вводимых данных и их квадратов с помощью предложения. . . $x_j \Pi + x^2 + x_{j+1} \Pi + x^2 + \dots$, а после ввода всех обрабатываемых данных получить \tilde{m}_1 делением содержимого регистра X на число N введенных данных, а $\tilde{\sigma}^2$ – несколькими несложными операциями. Конец вычислений в этом случае описывается программой: . . . $x_N \Pi + x^2 \div N = (\tilde{m}_1) X = X N = -$ ИП $\leftrightarrow = \div |N-1| = (\tilde{\sigma}^2)$.

В качестве примера рассмотрим задачу, которую пришлось решать леспромхозу в таежном крае. Требовалось определить число деревьев в таежном массиве площадью 45 кв. км при допустимой с вероятностью 0,999 погрешности не более 5 %. Для решения этой задачи в различных частях массива были выбраны 20 гектаров, в каждом из которых подсчет деревьев дал следующие результаты:

385	412	390	311	415	393	410	328	422	405
381	377	396	431	345	403	394	368	407	398

Так как сумма этих чисел равна 7771, то среднее число деревьев на гектар $\tilde{m}_1 = 7771/20 = 388,55$. А так как сумма квадратов всех чисел равна 3037271, то оценка дисперсии $\tilde{\sigma}^2 = (3037271 - 20 \cdot 388,55^2) = 939,421$, а оценка среднеквадратического отклонения $\tilde{\sigma} = \sqrt{939,421} = 30,65$. Используя взятый из табл. 6 коэффициент t для заданной доверительной вероятности 0,999 при числе измерений 20, находим, что истинное значение среднего числа деревьев на одном гектаре может отличаться от найденной оценки на $3,85 \cdot 30,65/20 = 26,39$ деревьев, что со-

отвечает предельной относительной погрешности $\tilde{\sigma}/\tilde{m}_1$ ($26,39 \cdot 100 : 388,55\% = 6,79\%$).

Так как возможная погрешность превышала допустимую, пришлось снова идти в лес и считать деревья. Получив данные еще с 20 гектаров в различных частях лесного массива

395	423	408	411	388	403	451	374	336	401
376	364	337	407	394	356	355	356	407	382

и повторив вычисления (которые легко может проверить читатель), нашли $\tilde{m}_1 = 387,375$; $\tilde{\sigma}^2 = 886,087$; $\tilde{\sigma} = 29,767$ для всех 40 гектаров и возможную ошибку при заданной доверительной вероятности $\tilde{\sigma}/\tilde{m}_1 = (3,55 \cdot 29,767 : 387,375 \times 40) \cdot 100 = 4,31\% < 5\%$. Полное число деревьев составило $387,375 \cdot 4500 = 1740000$.

Каких неожиданностей и неприятностей можно ожидать при решении других практических задач? При использовании микрокалькулятора с естественным представлением чисел для обработки больших массивов данных следует опасаться переполнения. Большинами погрешностями вычислений грозит массив данных, для которого выполняется соотношение $\tilde{m}_1/\tilde{\sigma} \gg 1$. Анализ погрешности, который читатель может провести самостоятельно в соответствии со сведениями, содержащимися в гл. 1, приводит к следующей формуле для относительной операционной ошибки (при вычислениях на восьмиразрядном микрокалькуляторе со стандартным представлением чисел):

$$\delta \tilde{\sigma}^2 / \tilde{\sigma}^2 \approx N \cdot 10^{-7} (1 + 2\tilde{m}_1^2 / \tilde{\sigma}^2).$$

Если $\tilde{m}_1/\tilde{\sigma} \geq 1000$, то может не оказаться ни одной верной цифры в значении $\tilde{\sigma}^2$.

Для уменьшения погрешности вычисления $\tilde{\sigma}^2$ целесообразно вычислять сумму обрабатываемых чисел и их квадратов при смещении начала отсчета на некоторое число x_0 , выбранное так, чтобы разность $x_j - x_0$ была бы поменьше, а вычисление этой разности "в уме" при вво-

де данных не было сложным. Тогда после вычисления сумм $x_j - x_0$ и $(x_j - x_0)^2$ оценки среднего определяются формулами

$$\tilde{m}_1 = \frac{\sum_{j=1}^N (x_j - x_0)}{N} + x_0;$$

$$\tilde{\sigma}^2 = \frac{\sum_{j=1}^N (x_j - x_0)^2 - [\sum_{j=1}^N (x_j - x_0)]^2/N}{N-1}.$$

Например, обрабатывая первые двадцать чисел в рассмотренной задаче, можно принять $x_0 = 300$ и использовать в вычислениях значения

85	112	90	11	115	93	110	28	122	105
81	77	96	131	45	103	94	68	107	98,

по которым найти $\sum_{j=1}^{20} (x_j - x_0) = 1771$ и $\sum_{j=1}^{20} (x_j - x_0)^2 = 174671$ и окончательно $\tilde{m}_1 = 1771 : 20 + 300 = 88,55 + 300 = 388,55$; $\tilde{\sigma}^2 = (174671 - 1771^2 : 20) : 19 = (174671 - 156822,05) : 19 = 939,41842$. В этом случае часто весьма ощутимо сокращается число вводимых значащих цифр обрабатываемого массива данных, что ускоряет вычисления, уменьшает вероятность случайных ошибок при вводе данных и снижает опасность переполнения.

При обработке больших массивов данных, существенно отличающихся по порядку от единицы, трудоемкость ввода чисел часто удается уменьшить "нормировкой" всего массива — умножением всех его элементов на такое число $M = 10^k$, при котором уменьшается количество вводимых знаков. Например, вместо ввода 0,000678; -0,000245; 0,00024 . . . вводят 678; -245; 240 . . . , приняв $M = 10^6$.

После вычисления статистических оценок преобразованного массива значение \tilde{m}_1 следует разделить на M , а $\tilde{\sigma}^2$ — на M^2 .

При статистической обработке данных оператор основное время затрачивает на их ввод, так как над ними выполняются чаще всего несложные операции. Поэтому при решении наиболее распространенных статистических задач с вводом данных в ЭВМ "вручную" она не имеет преимуществ по сравнению с микрокалькулятором, так как время ввода данных оказывается соизмеримым, а стоимость машинного времени микрокалькулятора несравненно ниже, чем у "больших" ЭВМ.

Если Вы используете программируемый микрокалькулятор, то программу вычисления среднего и дисперсии целесообразно составить так, чтобы получать "текущие" оценки после каждого ввода данных. Для этого можно воспользоваться формулами

$$\begin{aligned}\tilde{m}_{1i} &= \tilde{m}_{1(i-1)} + \frac{x_i - \tilde{m}_{1(i-1)}}{i}, \quad \tilde{\sigma}_i^2 = \frac{i-2}{i-1} \tilde{\sigma}_{(i-1)}^2 + \\ &+ \frac{(x_i - \tilde{m}_{1(i-1)})^2}{i},\end{aligned}$$

где i — номер числа, введенного последним, $\tilde{m}_{1(i-1)}$ и $\tilde{\sigma}_{(i-1)}^2$ — оценки среднего и дисперсии, вычисленные по $i-1$ предыдущим вводам, причем $\tilde{m}_{1i} = x_1$, $\tilde{\sigma}_1^2 = 0$, а x_i — введенный последним элемент обрабатываемого массива данных. Составленная по этим формулам программа на входном языке микрокалькулятора "Электроника Б3-21" имеет следующий вид.

Программа вычисления среднего и дисперсии

P2	2	P4	F2	C/P	↑	F2	—	↑	F4	÷	XY
X	←	F2	+	P2	F4	1	+	P4	3	—	1
+	÷	↑	F3	×	↑	→	+	P3	↑	БП	0

После ввода программы обрабатываемые числа x_i заносят в регистр X и нажимают клавишу С/П (после первого ввода – В/О и С/П). Обработка введенного числа оканчивается выводом на индикатор (и регистр 2) текущего значения \tilde{m}_{1i} , а в регистры Y и 3 – оценка $\tilde{\sigma}_i^2$ после ввода двух и более элементов массива. В регистре-счетчике 4 хранится число $i + 1$.

Распространенной задачей статистических исследований является оценка взаимосвязи между различными случайными величинами. Простейший вид такой взаимосвязи, заключающейся в линейной зависимости среднего значения случайной величины y_j от случайной величины x_j , оценивают по коэффициенту корреляции (соотносительности), эмпирически определяемому по формуле

$$\tilde{r}_{xy} = \frac{\sum_{j=1}^N (x_j - \tilde{m}_{1x})(y_j - \tilde{m}_{1y})}{\tilde{\sigma}_x \tilde{\sigma}_y (N-1)} = \frac{\sum_{j=1}^N x_j y_j - \tilde{m}_{1x} \tilde{m}_{1y}}{(N-1) \tilde{\sigma}_x \tilde{\sigma}_y}$$

Коэффициент корреляции может принимать по модулю значения от нуля до единицы. Если $\tilde{r}_{xy} = 0$, то между рассматриваемыми величинами взаимосвязь описанного типа отсутствует, при $\tilde{r}_{xy} = \pm 1$ анализируемые величины связаны линейным отношением вида $y = kx + b$, где k и b – некоторые коэффициенты. Отметим, что для получения правдоподобной оценки коэффициента корреляции необходимо обработать достаточно большие массивы данных, а определение доверительного интервала результата сложнее, чем для оценок среднего и дисперсии.

При определении оценки коэффициента корреляции на непрограммируемых микрокалькуляторах приходится

последовательно вычислять $\sum_{j=1}^N x_j y_j$, оценки \tilde{m}_{1x} , \tilde{m}_{1y} , $\tilde{\sigma}_x^2$ и $\tilde{\sigma}_y^2$, а затем подставлять их в приведенные форму-

лы. Вычисления часто удается упростить в связи с тем, что коэффициент корреляции не изменяется при любых линейных преобразованиях обрабатываемых массивов. Для обладателей программируемых микрокалькуляторов "Электроника Б3-21" приводим следующую программу.

Программа вычисления коэффициента корреляции

P2	X	XU	P7	F6	+	P6	F2	ПП	P7	P4	F3
P2	F7	ПП	P7	P5	F2	↑	F3	×	P7	←	F8
1	+	P8	F6	X	↑	F7	—	↑	F4	x#0	ВП
÷	↑	F5	÷	P7	C/П	↑	←	+	P3	←	XY
x^2	+	↑	←	F8	X	↑	F3	x^2	—	✓	B/O

После ввода этой программы регистры стека C1, C2, C4, C5 и регистр 6 ЗУПВ очищают, в регистр 8 записывают единицу, а в регистры X и Y соответственно вводят пары x_j и y_j , нажимая после каждого ввода пары этих чисел клавиши B/O и C/П. Выполнение программы (длящееся примерно 15 с) заканчивается выводом на индикатор и в регистр 7 оценки коэффициента корреляции. По содержанию остальных регистров ($P2 = C1 = \sum_{j=1}^N x_j$, $P3 = C4 = \sum_{j=1}^N y_j$,

$$P4 = \sqrt{N \sum_{j=1}^N x_j^2 - (\sum_{j=1}^N x_j)^2}, \quad P5 =$$

$$= \sqrt{N \sum_{j=1}^N y_j^2 - (\sum_{j=1}^N y_j)^2}, \quad P6 = \sum_{j=1}^N x_j y_i, \quad P8 = N+1, \quad C3 =$$

$$= \sum_{j=1}^N x_j^2, \quad C5 = \sum_{j=1}^N y_j^2, \quad C6 = \sum_{j=1}^N x_j \cdot \sum_{j=1}^N y_j)$$

нетрудно в обычном режиме вычислить оценки \tilde{m}_{1x} , \tilde{m}_{1y} , $\tilde{\sigma}_x^2$ и $\tilde{\sigma}_y^2$ (для тренировки составьте соответствующие выражения на входном языке своего микрокалькулятора).

Для случайных величин, вводимых в определенном порядке, вычисляют коэффициент корреляции между их

значениями, разделенными определенным интервалом (который, например, может соответствовать интервалу времени, разделяющего получение данных во многих задачах). В этом случае можно использовать формулу для оценки коэффициента корреляции между соседними отсчетами x_j и x_{j+1} , подставляя их вместо x_j и y_j (суммирование выполняется до $N - 1$); при вычислении коэффициента корреляции, между отсчетами, разделенными двумя "шагами" – значения x_j и x_{j+2} (суммирование до $N - 2$) и т.д. Набор вычисленных подобным образом значений $r_x(h)$, где h – шаг, задающих автокорреляционную функцию изучаемого случайного процесса, используют при изучении его закономерностей.

В заключение отметим, что результаты физических процессов, повторяющихся с одинаковой точностью (например, результаты измерений с помощью измерительных приборов), распределены, как правило, по нормальному закону.

2.5. Блуждание в тумане, или оптимизация параметров

Решение задачи оптимизации параметров можно показать на следующих примерах.

Однажды наш читатель Иванов отправился в лес на прогулку. Ему не повезло – туман сгустился настолько, что он заблудился и в некоторой растерянности остановился на холме у топографического знака. Помня, что его дом стоит в долине, он решил, что если спускаться все время вниз, то обязательно попадет домой. Чтобы не блуждать напрасно, решил идти вначале точно на север до самого низкого места в этом направлении. Затем повернуть на запад или восток (в направлении снижения местности) и снова найти самое пониженное место. Меняя таким образом направления поиска, Иванов надеялся, в конце концов, попасть в долину, где стоял его дом. Но вместо дома он скоро попал в глубокий овраг, по дну которого про-

текал ручей. Учитывая, что вода стекает вниз по наибольшему уклону в долину, Иванов отправился вдоль ручья, считая такой спуск скорейшим.

Иванов успешно спустился по оврагу, но затем попал в заболоченную долину, вокруг которой местность повышалась. Пришлось выбираться из нее по случайно выбранному направлению. Пройдя двести шагов, он снова нашел направление скорейшего спуска и, обойдя засеянное поле, вскоре очутился дома.

Иванов торжествовал, но его самоуверенность исчезла, когда он проложил пройденный путь на топографической карте. Оказалось, что, выйдя из начальной точки 0 (рис. 19), он удачно выбрался из оврага A и долины B, но, не остановившись случайно в пункте C, прошел бы гребень холма и спустился совсем в другую долину. Повезло ему также и со случайным выбором направления обхода поля (в другом направлении оно тянулось на километры). Не будь этих случайностей, он не так скоро нашел бы свой дом.

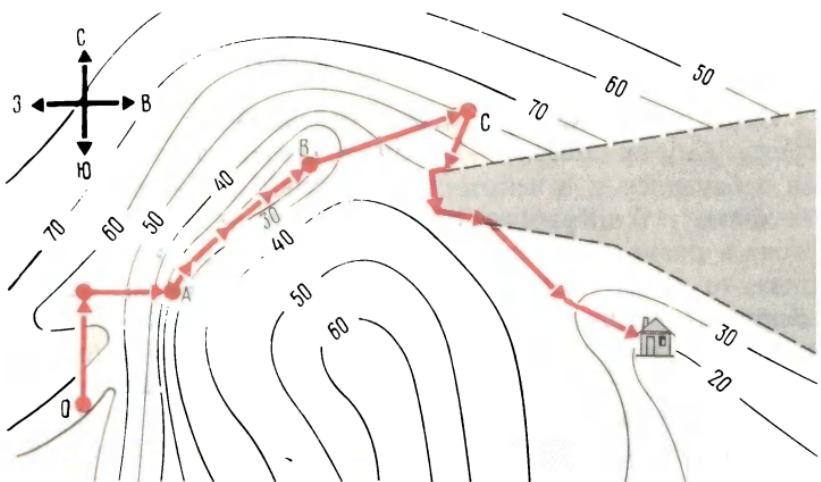


Рис. 19. Топографический план окрестностей дачи Ивановых

С аналогичной задачей мы сталкиваемся, когда нужно, например, изготовить бак для воды емкостью в один кубометр, а листового железа имеется всего один лист площадью $4,9 \text{ м}^2$ и следует найти наиболее экономное решение: Для кубического бака (даже без крышки) при ширине a , длине b и высоте h , равных одному метру, требуетсѧ листового железа $S = ab + 2h(a + b) = 5 \text{ м}^2$, а имеется только $4,9 \text{ м}^2$.

Для того чтобы хватило имеющегося железа, попробуем уменьшить высоту бака до $h = 0,9 \text{ м}$. В этом случае при $a = b = \sqrt{1/h} = 1,05409 \text{ м}$ требуемая площадь железа $S = 4,9062 \text{ м}^2$ действительно меньше, чем в первом случае. Остается выбрать способ решения задачи.

Согласно исходным данным сформулируем его как поиск таких ширины a , длины b и высоты $h = 1/ab$, при которых площадь бака $S = ab + 2(a + b)/ab$ имела бы наименьшее значение. При этом значения a и b соответствовали бы этой площади, а высота определялась по заданной емкости бака.

Используя (как и в задаче блужданий в лесу) метод координатного спуска к минимуму функции S , составим бланк для записи результатов вычислений и вооружимся микрокалькулятором. Выбрав начальные значения $a = b = 0,5$ начнем поочередно изменять переменные до достижения минимального значения S , пока не получим искомый результат: $a = b = 1,26 \text{ м}$ и $h = 1,26^{-2} = 0,63 \text{ м}$, которому соответствует требуемая площадь железа $S = 4,762 \text{ м}^2$.

Однако, определяя размеры бака, мы не учитывали размеры листа железа ($2 \times 2,45 \text{ м}^2$), при которых раскроить его оказывается невозможным. Тем самым не учитывались практические ограничения задачи.

Разметив примерную раскройку листа (рис. 20), составим (с учетом потерь железа $\Delta = 1 \text{ мм}$ на каждый разрез) систему ограничений неравенств, при удовлетворении которых решается задача: $S \leq 4,9 \text{ м}^2$; $2b + \Delta \leq 2,45 \text{ м}$;

$b + 2h + 2\Delta \leq 2,45$ м; $a + h + \Delta \leq 2$ м. Отсюда при $h = 1/ab$ следуют требования: $b + 2/ab \leq 2,449$ м; $a + 1/ab \leq 1,999$ м; $b \leq 1,2245$ м.

Вычислив с помощью микрокалькулятора для различных значений b значения a , при которых полученные неравенства обращаются в равенства, вычертим в плоскости координат a и b соответствующие граничные линии, включая и линию, ограничивающую допустимую площадь железа $S = 4,9$ м². Оказывается, что хотя область допустимых решений и мала (рис. 21), бак все-таки можно изготовить. Выбрав в этой области значения $b = 1,2$ м и $a = 1,4$ м, вычислим $h = 0,596$ м, $V = 1,00128$ м³ и $S = 4,775$ м². Полученные результаты нас вполне устраивают. Более того, при $a + h + 2\Delta = 1,997 \leq 2$ м² и $b + 2h + 2\Delta = 2,394 \leq 2,4$ м остается еще кусок железа площадью, превышающей квадратный дециметр.

Вернемся к задаче поиска оптимального пути и проверим возможность решения этой задачи скорейшим спуском. Так как наибольшему уклону местности соответствует наибольшее отрицательное значение производной от высоты по направлению, найдем прежде всего выражения для производных по координатам

$$S'_a = dS/da = b - 2/a^2; S'_b = dS/db = a - 2/b^2,$$

а затем и производную по направлению скорейшего спуска (антиградиент)

$$\nabla = -\sqrt{(S'_a)^2 + (S'_b)^2}.$$

Заготовив вычислительный бланк, заметим, что в силу симметрии функции $S(a, b)$ относительно обеих переменных задача сводится к поиску минимума функции $S = x^2 + 4/x$ одной переменной $x = a = b$. Выполнив вычисления, приходим к тому же результату, что и раньше.

Построим с помощью микрокалькулятора "топографическую карту" (рис. 22) функции $S(a, b)$. Симметрия функции относительно координатных осей и наличие

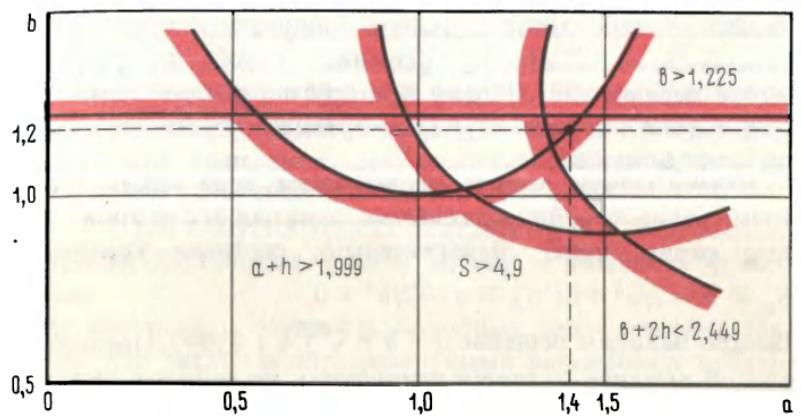


Рис. 20. Нелинейные ограничения в задаче об изготовлении бака

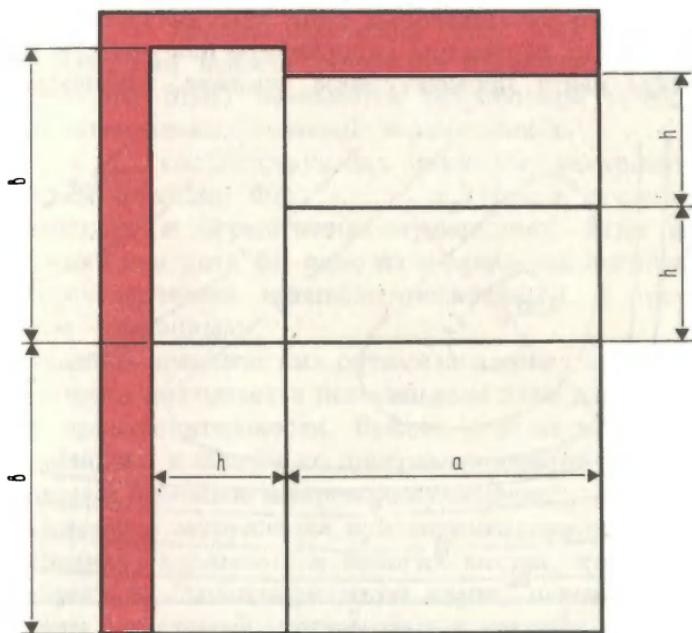


Рис. 21. Раскройка листа железа для изготовления бака

только одного минимума, что видно из этой карты, способствуют быстрому решению задачи по функции одной переменной. Однако при другом выборе начальной точки (вне оси симметрии) скорейший спуск оказывается более длительным.

Задачу можно решить аналитически, если учесть, что в точке минимума или максимума функции все ее производные равны нулю. Действительно, составив уравнения $S'_a \equiv b - 2/a^2 = 0$; $S'_b \equiv a - 2/b^2 = 0$,

быстро находим решение $a = b = \sqrt[3]{2} = 1,25992$. Последний способ решения является наилучшим, но решение системы уравнений, полученной приравниванием нулю производных, в большинстве случаев более сложное, чем решение исходной задачи отыскания минимума одного уравнения.

При решении рассмотренных выше задач мы стремились так изменить их параметры (координаты или размеры бака), чтобы найти промежуточное решение, отвечающее

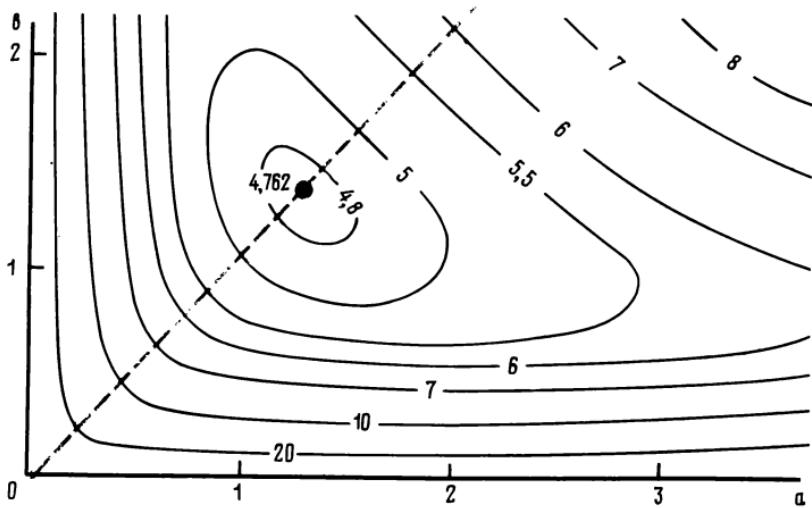


Рис. 22. "Топографическая карта" функции $S(a, b)$

определенному критерию качества оптимальности (близости к дому или возможности построить бак из имеющихся запасов железа). Процесс решения подобных задач называют *оптимизацией параметров*, а функциональные зависимости (например, зависимости высоты местности от координат или площади бака от его размеров), минимум или максимум которых соответствует оптимальным значениям параметров и цели задачи, — *целевыми функциями*.

Характерная особенность подобных задач заключается в том, что число m оптимизируемых переменных превышает число r ограничений-равенств, связывающих эти переменные. По этим равенствам можно найти значения r переменных, а оптимальные значения остальных $n = m - r$ переменных приходится подбирать методами *математического программирования*. Этот раздел математики (не имеющий непосредственного отношения к программированию ЭВМ) занимается разработкой методов поиска оптимальных значений переменных $x_1, \dots, x_n, \dots, x_{n+r}$, соответствующих минимуму неотрицательной целевой функции $\Phi(x_1, \dots, x_n)$ при r ограничениях-равенствах и ограничениях-неравенствах. Если целевая функция или хотя бы одно из ограничений нелинейны, то программирование называют *нелинейным*, в противном случае — *линейным*.

Решение практических оптимизационных задач громоздко и часто оказывается непосильным даже для ЭВМ высокой производительности. Вместе с этим многие задачи нелинейного и линейного программирования успешно решимы с помощью микрокалькуляторов.

Основные затруднения при оптимизации параметров — медленная сходимость в пологих местах, крутые гребни и овраги на "топографической карте" целевой функции, долины с местными минимумами и, наконец, ограничения, подобные полю. При решении задач мы использовали на-

иболее известные методы нелинейного программирования – координатный и скорейший спуски, а также случайный поиск выхода из затруднительного положения. Характерная особенность подобных методов заключается в последовательности итераций, на каждой из которых выбирают определенное направление и ищут минимум функции на этом направлении, после чего переходят к следующей итерации.

Одномерный поиск имеет самостоятельное значение при поиске минимума (минимизации) функции одной переменной. При таком поиске вначале определяют интервал, в котором находится оптимальное (соответствующее минимуму функции) значение аргумента, а затем сокращают ширину этого интервала до заданного значения ϵ , определяющего требуемую точность вычисления оптимального значения аргумента.

При отыскании начального интервала оптимума задаются исходным значением аргумента x_0 и для ряда значений x_i с постоянным или возрастающим шагом вычисляют значения минимизируемой функции $\Phi(x_i)$. При приближении к минимуму эти значения уменьшаются, но при его прохождении начинают возрастать. При увеличении функции на k -м шаге начальным выбирают интервал $[x_{k-2}; x_k]$, так как при выборе начальным интервала $[x_{k-1}; x_k]$ может возникнуть, как следует из рис. 23, а, ошибка.

Ширину начального интервала оптимума сокращают различными способами, в частности, известным нам по решению уравнений методом половинного деления. Однако при поиске минимума (в отличие от поисков корня) значения функции на границах сокращаемого интервала совпадают по знаку, что усложняет процедуру минимизации функции.

При одномерном поиске иногда используют золотое сечение $\varphi = 1,6183$, выбирая начальное значение x_0 и после-

довательно вычисляя $x_i = \varphi x_{i-1}$ и $\Phi(x_i)$ до выполнения условия $\Phi(x_{k-2}) > \Phi(x_{k-1}) < \Phi(x_k)$. Полученный начальный интервал с границами $x_{\text{н}} = x_{k-2}$ и $x_{\text{в}} = x_k$ делят на три части в точках $x_{10} = x_{\text{н}} + 0,38\Delta$ и $x_{20} = x_{\text{н}} + 0,62\Delta$, где $\Delta = x_{\text{в}} - x_{\text{н}}$; $0,38 \approx 1/\varphi^2$ и $0,62 \approx 1/\varphi$. Если $\Phi(x_{10}) \geq \Phi(x_{20})$, то оптимум находится в интервале $[x_{10}; x_{\text{в}}]$, в противном случае — в интервале $[x_{\text{н}}; x_{20}]$. Подобные операции (рис. 23, б) повторяют до выполнения условия $\Delta \leq \epsilon$. Так, при минимизации функции $\Phi(x) = x^2 + 4/x$ методом золотого сечения, выбрав, например, $x_0 = 0,5$ и вычислив $\Phi(0,5) = 8,25$, находят далее $x_1 = \varphi x_0 = 0,81$; $\Phi(x_1) = 5,594371$; $x_2 = \varphi x_1 = 1,3122$; $\Phi(x_2) = 4,770184$; $x_3 = \varphi x_2 = 2,125764$; $\Phi(x_3) = 6,400548$. Следовательно, оптимум находится в интервале $[0,81; 2,125764]$, и, сократив этот интервал, например до $\Delta \leq \epsilon = 0,001$, получают $x_{\text{опт}} = 1,259$.

При использовании программируемых микрокалькуляторов удобны последовательные равномерные методы, при которых вычислением функции в точках $x_i = x_{i-1} + h$ находят одну границу оптимума, затем изменяют знак и уменьшают шаг h до отыскания другой границы, повторяя подобные операции до сокращения интервала оптимума

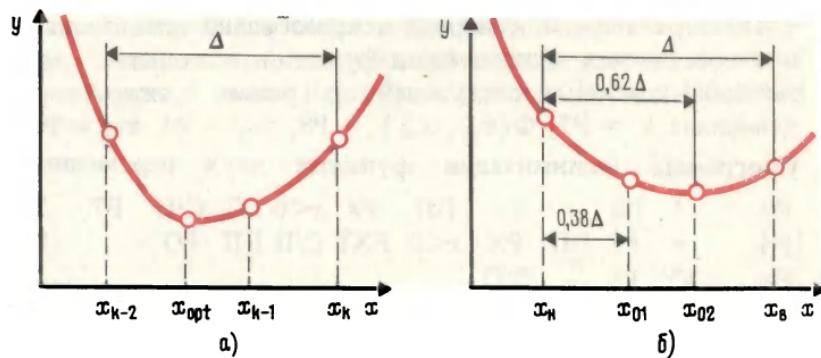


Рис. 23. Поиск интервала оптимального значения аргумента

$\Delta \leq \epsilon$. Этот метод реализован в следующей программе ($\epsilon^2 = P2$, $x_0 = P4$, $-ah = P7$, $\Phi(x_0) = P8$).

Программа минимизации функции одной переменной

F7	a	\div	$/-$	$\cdot P7$	F4	\uparrow	F7	$+$	P4	\dots	\uparrow
F8	XY	P8	$-$	$x < 0$	\uparrow	F7	x^2	\uparrow	F2	$-$	$x < 0$
P0	C/P										

В программе букву a заменяют выбранным однозначным (от 2 до 9) делителем шага, а многоточие – операторами вычисления минимизируемой функции. После выполнения программы (пуск нажатием клавиш В/О и С/П) вычисленные значения x_{opt} и Φ_{\min} хранятся в регистрах 4 и 8. Функцию $\Phi(x) = x^2 + 4/x$ (в программу записывают $4 \div 1/x XY x^2 +$) при $a = 2$, $h = x_0 = 0,5$ и $\epsilon = 0,001$ эта программа минимизирует до значения $\Phi_{\min} = 4,762206$ при $x_{\text{opt}} = 1,258789$ за 1 мин 30 с.

Если операторы этой программы, начиная с F7 по адресу 31, заменить операторами F7 C/P БП P0, то вычисления будут прерываться после каждой итерации с высвечиванием значения аргумента на одной из границ сокращенного интервала оптимума. Это позволяет контролировать ход минимизации функции и, при необходимости, изменять текущие значения x и h .

Поитерационный контроль целесообразно использовать и в программах минимизации функций нескольких переменных, подобных следующей программе с исходными данными: $h = P7$, $\Phi(x_{01}, x_{02}) = P8$, $x_{01} = P4$, $x_{02} = P5$.

Программа минимизации функции двух переменных

F7	\uparrow	F4	$+$	P4	ПП	РХ	$x < 0$	P0	C/P	F7	\uparrow
F5	$+$	P5	ПП	РХ	$x < 0$	FXY	C/P	БП	P0	\dots	\uparrow
F8	XY	P8	$-$	B/O							

После пуска этой программы нажатием клавиш В/О и С/П она находит границу интервала оптимума по переменной x_1 , а после нажатия клавиши С/П – по перемен-

ной x_2 . Последовательно нажимая только эти клавиши, можно сокращать интервал оптимума по каждой из переменных или по обеим вместе, но перед каждым пуском программы следует изменить на обратный знак шага h и уменьшить его абсолютную величину. Эта программа, в частности, позволяет за 15...20 итераций найти минимум функции $\Phi = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$, обычно используемой в качестве тестовой при проверке оптимизационных программ на больших ЭВМ.

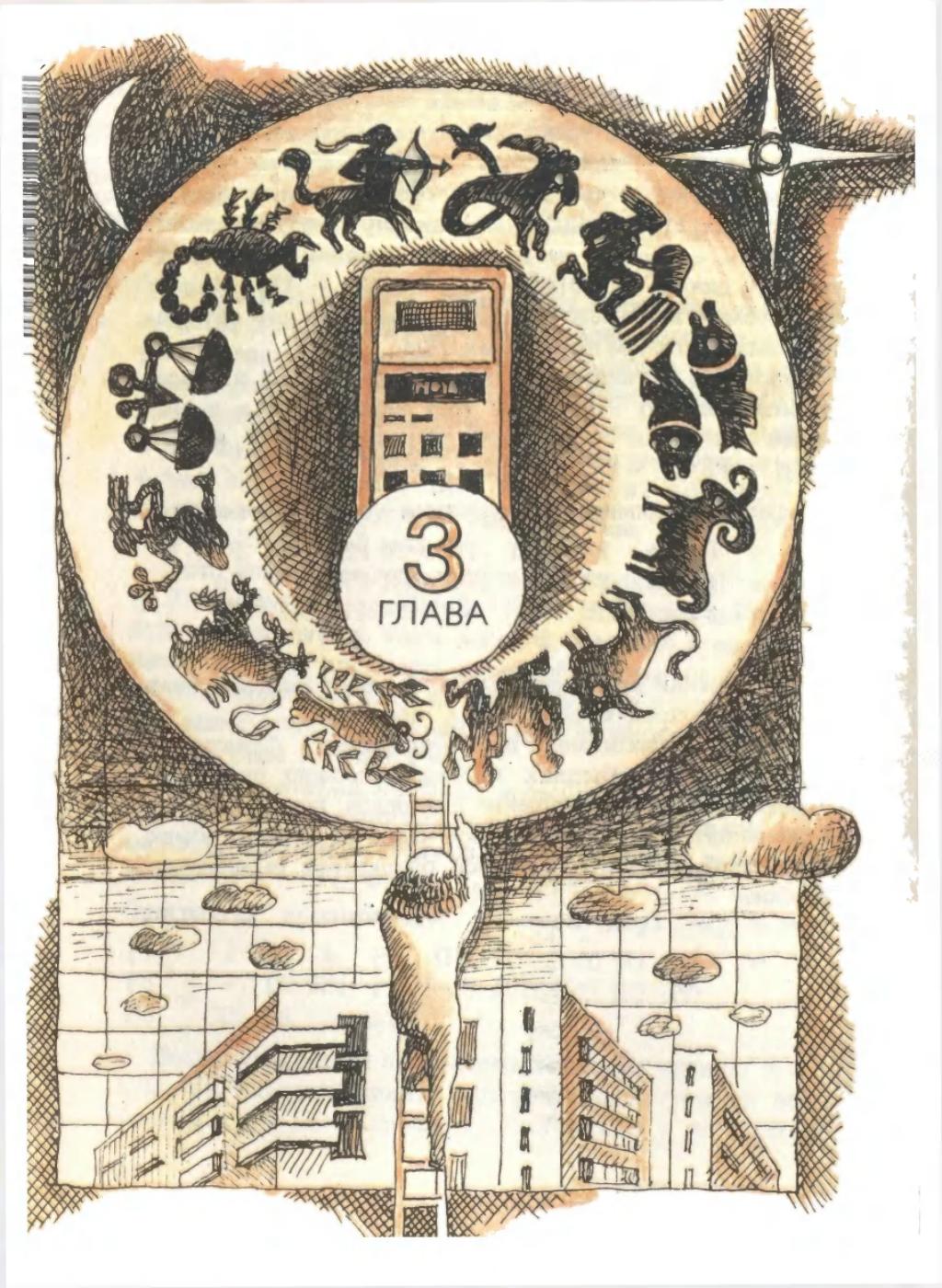
Аналогично организованы вычисления границ интервалов оптимума по каждой из трех переменных и в следующей программе с исходными данными: $h = P7$, $\Phi(x_{01}, x_{02}, x_{03}) = P8$, $x_{01} = P4$, $x_{02} = P5$, $x_{03} = P6$.

Программа минимизации функции трех переменных

F7	↑	F4	+	P4	ПП	P7	$x < 0$	РО	C/П	F7	↑
F5	+	P5	ПП	P7	$x < 0$	FXY	C/П	F7	↑	F6	+
P6	ПП	F7	$x < 0$	P÷	C/П	...	↑	F8	XY	P8	-
B/O											

При выполнении этой программы текущие значения всех переменных сохраняются в "своих" регистрах.

Уровень автоматизации при минимизации функций с помощью двух последних программ можно повысить, автоматизировав и сокращение интервала, но в этом случае уменьшится и так небольшое число свободных ячеек программной памяти для записи операторов вычисления функции.



3

ГЛАВА

НАУКА И ЖИЗНЬ

**ПОДАРОК МОЛОДЫМ ХОЗЯЙКАМ
ВАШ ДЕНЬ РОЖДЕНИЯ И БИОРИТМЫ
ПОКУПАТЬ ЛИ ШКАФ?
"РЕГАТА" НА КЛАВИШАХ
БЕРЕГИТЕ АВТОМОБИЛЬ!**

3.1. Подарок молодым хозяйствам

Очень часто перед праздником Вы, читатель, решаете следующую задачу: как на определенную сумму денег купить наибольшее количество подарков, но при этом однотипных и с учетом их массы и вместимости вашей сумки. Попробуем решить с Вами поставленную задачу, используя микрокалькулятор.

Основную цель (купить, например, на 20 рублей побольше чашек и тарелок) отобразим в виде условий функции Φ двух переменных: x_1 , равной числу чашек массой $m_1 = 0,4$, и x_2 , равной числу тарелок массой $m_2 = 1,0$. В этом случае целевую функцию Φ можно записать $\Phi = 0,4x_1 + x_2$.

Ограничения задачи, связанные с необходимостью приобретения чашек и тарелок, опишем неравенствами $x_1 \geq 1$ и $x_2 \geq 1$. Наиболее существенное ограничение, связанное с отведенной для покупки чашек и тарелок суммой денег, — неравенством $x_1 + 3x_2 \leq 20$, так как чашка стоит 1 руб., а тарелка — 3 руб. Чисто техническое, но не менее существенное ограничение, определяется небольшим объемом (10 куб. единиц) сумки для покупок. С учетом объемов чашки (0,4 куб. единицы) и тарелки (0,8 куб. единиц) это ограничение выражается неравенством $0,6x_1 + 0,8x_2 \leq 10$.

Поскольку в нашей задаче переменные только две, то условие задачи можно отобразить на рисунке. Ограничения $x_1 \leq 1$ и $x_2 \leq 1$ легко учесть линиями, параллельными координатным осям, отметив при этом штриховкой границы запрещенной области, в которой не выполняются заданные ограничения (рис. 24).

Для ограничения $x_1 + 3x_2 \leq 20$ отметим на координатных осях точками $x_1 = 20$ и $x_2 = 20/3 \approx 6,67$ и проведем через них прямую. Подобным образом отобразим и последнее из ограничений, получив область допустимых решений в виде четырехугольника $ABCD$, любая точка внутри и на границах которого удовлетворяет всем заданным ограничениям. Остается найти такую из них, в которой целевая функция будет максимальной. Интуиция подсказывает, что такая точка лежит в одной из вершин допустимой области, но нужно доказать это предположение.

Легко заметить, что увеличение целевой функции приводит к смещению соответствующей ей прямой параллельно самой себе. Поэтому снабдим ее "колесиками" (рис. 24), убедившись, что во всех точках этой прямой значения целевой функции одинаковы. Если "покатить" целевую функцию вверх, то убедимся, что она максимальна

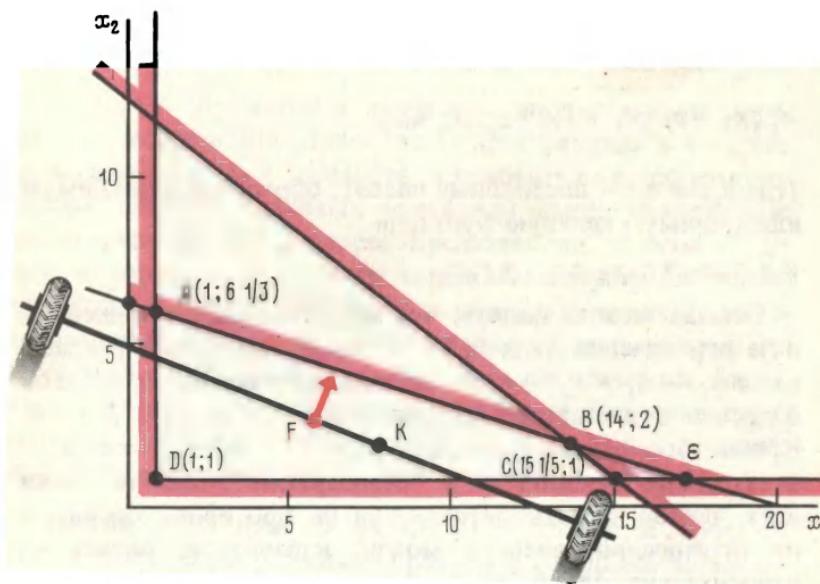


Рис. 24. Графическая модель в задаче о покупке мяса

в вершине допустимой области с координатами $x_1 = 14$ и $x_2 = 2$, что и соответствует числу чаек и тарелок, при котором полностью используется объем сумки и выделенные на покупку деньги.

Графическое представление задачи позволяет сделать несколько выводов общего характера. Прежде всего отметим, что не каждая задача подобного типа даже с двумя переменными имеет решение — допустимая область может оказаться неограниченной сверху и, кроме того, исходные ограничения могут оказаться противоречивыми. Однако после решения этой задачи остается неясным, как решать подобные задачи при большем числе переменных.

В общем случае эта задача сводится к поиску оптимальных значений неотрицательных переменных x_1, x_2, \dots, x_n , которые, удовлетворяя ограничениям

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &\leq a_1; \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &\leq a_2; \\ \dots & \\ a_{r_1}x_1 + a_{r_2}x_2 + \dots + a_{rn}x_n &\leq a_r \end{aligned}$$

(где a_{ij} и a_i — постоянные числа), обращают в максимум или минимум целевую функцию

$$\Phi = c_1x_1 + c_2x_2 + \dots + c_nx_n.$$

Отсюда можно понять, что встретившиеся в нашей задаче ограничения вида $x_i \geq b_i$ легко заменить неравенствами $x_i - b_i = x'_i \geq 0$, а для поиска максимума целевой функции Φ достаточно минимизировать функцию $\Phi' = -\Phi$. Кроме того, ограничения вида $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geq a_i$ легко превратить в "стандартные", сменив знаки всех членов и знак неравенства на противоположные, а от ограничений-равенств можно избавиться, решив их относительно одной из переменных и исключив ее при минимизации из остальных ограничений.

Хотя при числе переменных $n > 2$ графическое решение оказывается невозможным (в этом случае целевая функция и ограничения отображаются гиперплоскостями в пространстве n координат, а допустимая область превращается в "многогранник"), можно отметить, что вывод о достижении целевой функцией максимума в одной из вершин допустимой области (или на ребре, параллельном гиперплоскости целевой функции) остается справедливым. Поэтому при решении задач линейного программирования, к которым относится и наша задача, достаточно вычислять целевую функцию лишь в вершинах допустимой области, выбрав ту из них, в которой достигается искомый минимум или максимум.

При определении координат вершин все ограничения-неравенства преобразуют в равенства вида $a_{i_1}x_1 + a_{i_2}x_2 + \dots + a_{i_n}x_n + y_i = a_i$ введением дополнительных неотрицательных переменных y_i , определяющих недоиспользование запаса a_i соответствующего i -го ресурса.

В общем случае после введения дополнительных переменных число неизвестных становится равным $n + r$, что превышает число r равенств. Следовательно, сформированная система линейных уравнений имеет бесконечное число решений, но интерес представляют только те из них, которые соответствуют вершинам допустимой области. В этих вершинах лишние (их называют *свободными*) n переменных равны нулю, что позволяет решить систему относительно остальных r переменных, называемых *базисными*. Если после решения системы все базисные переменные окажутся неотрицательными, то такое решение называют *опорным*.

Замена свободной переменной на базисную соответствует в опорном решении переходу в соседнюю вершину допустимой области. Решение является опорным, если все коэффициенты a_i в ограничениях неотрицательны,

и оптимальным (при минимизации функции), если все коэффициенты целевой функции положительны.

При большом числе переменных требуемые преобразования выполняют в соответствии с формализованным алгоритмом, называемым симплекс-методом. При его использовании условия задачи, приведенные к стандартной форме, отображают таблицей (матрицей) следующего вида:

y	$-x_1$	$-x_2$...	$-x_n$	1
y_1	a_{11}	a_{12}	...	a_{1n}	a_1
y_2	a_{21}	a_{22}	...	a_{2n}	a_2
...
y_r	a_{r1}	a_{r2}	...	a_{rn}	a_r
Φ	$-c_1$	$-c_2$...	$-c_n$	

В первую строку подобной симплекс-таблицы записывают с отрицательным знаком символы оптимизируемых переменных (и единицу в последнем столбце), в следующих r строках — символы дополнительных переменных y (первый столбец), коэффициенты a_{ij} с учетом отрицательного знака переменных и, в последнем столбце, свободные члены a_i . В нижней Φ -строке записывают с обратным знаком коэффициенты целевой функции, сохраняя свободной последнюю клеточку таблицы для последующей записи минимизируемых значений целевой функции.

Если среди свободных коэффициентов есть отрицательные, то опорное решение находят согласно следующему алгоритму:

1. В строке, содержащей наибольший по модулю отрицательный свободный член a_k , выбирают отрицательный элемент a_{ks} и s -й столбец называют *разрешающим*.

2. Для каждого элемента этого столбца вычисляют неотрицательные отношения $a_i/a_{is} \geq 0$ и выбирают наименьшее из них, если $a_{is} \geq 0$, или наибольшее в противном случае. Соответствующие элемент a_{ks} и k -ю строку называют разрешающими.

3. Выполняют преобразование таблицы (с записью его результатов в новую таблицу), называемое шагом модифицированного жорданова исключения:

3.1. Разрешающий элемент a_{ks} заменяют на $a'_{ks} = 1/a_{ks}$.

3.2. Элементы разрешающей строки вычисляют по формуле $a'_{kj} = a_{kj}/a_{ks}$.

3.3. Элементы разрешающего столбца вычисляют по формуле $a'_{is} = -a_{is}/a_{ks}$.

3.4. Остальные элементы новой таблицы (включая свободные члены и элементы Φ -строки) вычисляют по формуле $a'_{ij} = a_{ij} - a_{is}a_{kj}/a_{ks}$.

3.5. Меняя местами символы переменных в s -м столбце и k -й строке.

4. При необходимости повторяют пп. 1–3 до получения неотрицательными всех свободных членов. Если в строке с отрицательным свободным членом все коэффициенты положительны, то задача неразрешима в связи с противоречивостью исходных ограничений.

Симплекс-таблица упрощается, если в строке с нулевым свободным членом все коэффициенты неотрицательны – в этом случае вычеркивают все столбцы, содержащие положительные из этих коэффициентов, и саму строку.

Поиск оптимального решения, соответствующего минимуму целевой функции, обеспечивается следующим алгоритмом:

1. Выбирают разрешающий s -столбец с наибольшим по модулю положительным элементом Φ -строки (здесь и далее без учета последнего элемента, отображающего значение целевой функции).

2. Разрешающую строку и элемент выбирают, как и при поиске опорного решения.
 3. Выполняют шаг модифицированного жорданова исключения для разрешающего элемента, как и при поиске опорного решения.
 4. При необходимости повторяют выполнение пп. 1–3 до получения неположительными всех элементов Φ -строки, кроме a_{r+1} .
- При выполнении п. 3 целесообразно прежде всего вычислить значения свободных членов и элементов Φ -строки, так как для получения оптимального решения остальные элементы вычислять не нужно. Значения переменных, оставшихся в первой строке оптимальной симплекс-таблицы, равны нулю, а оставшихся в первом столбце – значениям свободных членов. Дополнительные переменные, сохранившиеся в первом столбце, определяют недоиспользование соответствующих ресурсов.

Попытаемся решить симплекс-методом нашу задачу. Приведя неравенства $x_1 \geq 1$ и $x_2 \geq 1$ к виду $x_1 - 1 = x'_1 \geq 0$ и $x_2 - 1 = x'_2 \geq 0$, получаем

$$\begin{aligned}x'_1 + 3x'_2 + y_2 &= 20 - (1 + 3) = 16; \\0,6x'_1 + 0,8x'_2 + y_2 &= 10 - (0,6 + 0,8) = 8,6.\end{aligned}$$

Следовательно, в стандартной постановке задача сводится к поиску оптимальных значений x'_1 и x'_2 , удовлетворяющих равенствам

$$\begin{aligned}y_1 &= -x'_1 - 3x'_2 + 16; \\y_2 &= -0,6x'_1 - 0,8x'_2 + 8,6\end{aligned}$$

и минимизирующих целевую функцию

$$\Phi' = -\Phi = -0,4x'_1 - x'_2,$$

где за ненадобностью не учтен общий множитель 1,4.

Симплекс-таблица исходных условий соответствует опорному решению, так как свободные члены положительны.

y	$-x'_1$	$-x'_2$	1
y_1	1	3	16
y_2	0,6	0,8	8,6
Φ'	0,4	1	—

Выбрав на пересечении столбца с наибольшим положительным элементом Φ -строки и строки с наименьшим отношением a_i/a_{is} разрешающий элемент $a_{12} = 3$, выполняем с помощью микрокалькулятора шаг модифицированного жорданова исключения:

—	$-x'_1$	$-y_1$	1
x'_2	0,3333333	0,3333333	5,3333333
y_2	0,3333333	-0,2666666	4,3333333
Φ'	0,06666667	-0,3333333	-5,3333333

В столбце с положительным элементом Φ -строки выбираем разрешающим элемент $a_{21} = 0,3333333$, соответствующий минимуму отношений a_i/a_{is} , и вычисляем

—	$-y_2$	$-y_1$	1
x'_2			1
x'_1			13
Φ'	-0,2	-0,28	-6,2

Эта таблица соответствует оптимальному решению $x_1^{\text{opt}} = x'_2 + 1 = 14$; $x_2^{\text{opt}} = x'_2 + 1 = 2$; $\Phi'_{\max} = 6,2$.

С помощью симплекс-метода и микрокалькулятора можно составить дневной рацион питания при минимальной стоимости. Исходными данными являются: нормы суточной потребности человека в основных питательных веществах и калорийности пищи (табл. 7), а также химический состав и калорийность основных продуктов питания (табл. 8). В качестве иллюстрации покажем это на примере семьи Ивановых, состоящей из трех человек.

По этим данным для семьи из трех человек возможная минимальная суточная потребность оценивается в 300 г белков, 300 г жиров, 1500 г углеводов и 10 000 ккал. Зная цены основных продуктов питания, включим в первоначальный расчет только хлеб, молоко, гречневую крупу, растительное масло, говядину, макароны и картофель (табл. 7, А) при выборе в качестве минимизируемой функции стоимость рациона в рублях.

В соответствии с алгоритмом симплекс-метода выберем разрешающий элемент $a_{21} = -7$ и, выполнив с помощью микрокалькулятора шаг преобразования (табл. 7, Б), получим опорное решение, так как все свободные члены оказались положительными. Выбрав разрешающим элемент $a_{14} = 6365,1$, получим оптимальное решение (табл. 7, В), соответствующее суточному рациону для семьи из трех человек: 0,27 кг растительного масла и 6,38 кг хлеба при стоимости всего 1 р. 72 к. Проверка показывает, что этот рацион действительно обеспечивает ежедневно по 300 г жиров и белков и даже дает избыток углеводов (2502 г) и теплотворной способности (14 312 ккал).

Однако столь изящное решение задачи напоминает нам, что человеку нужны еще витамины. Используя данные о содержании витаминов, составим более сложную исходную симплекс-таблицу (табл. 8), а для облегчения вычислений составим следующую программу для микрокалькулятора "Электроника Б3-21".

Таблица 7

Минимизация стоимости суточного рациона

<i>A</i>	Хлеб $-x_1$	Молоко $-x_2$	Крупа гр. $-x_3$	Маслораст. $-x_4$	Говядина $-x_5$	Макароны $-x_6$	Картофель $-x_7$	1
Белки, y_1	-47	-33	-88	0	-190	-94	-14	-300
Жиры, y_2	-7	-35	-23	-948	-95	-8	0	-300
Углев., y_3	-392	-44	-634	0	0	-712	-190	-1500
ккал, y_4	-1870	-640	-3170	-8820	-1660	-3380	-840	-10000
Φ	-0,2	-0,2	-0,6	-1,62	-2	-0,6	-0,1	

<i>B</i>	Жиры $-y_1$	Молоко $-x_2$	Крупа гр. $-x_3$	Масло рабт. $-x_4$	Говядина $-x_5$	Макароны $-x_6$	Картофель $-x_7$	1
Белки, y_1	-6,7143	202	66,429	6365,1	447,86	-40,286	-14	1714,3
Хлеб, x_1	-0,14286	5	3,2857	135,43	13,571	1,1429	0	42,857
Углев., y_3	-56	1916	654	53088	5320	-264	-190	15300
ккал., y_4	-267,14	8710	2974,3	244431	23719	-1242,9	-840	70143
Φ	-0,0286	0,8	0,0571	25,466	-0,714	-0,3714	-0,1	8,5714

Окончание табл. 7

<i>B</i>	Жиры $-y_4$	Молоко $-x_1$	Крупа гр. $-x_3$	Белки $-y_1$	Говядина $-x_5$	Макароны $-x_6$	Картофель $-x_7$	1
Масло, x_4								0,2693
Хлеб, x_1								6,3830
Углеводы, y_3								1002,1
ккал., y_4								4311,6
Φ	-0,0017	-0,0082	-0,208	-0,004	-1,078	-0,21	-0,44	1,72

Симплекс-таблица, учитывющая содержание витаминов

<i>A</i>	Хлеб $-x_1$	Масло раст. $-x_2$	Морковь $-x_3$	Картофель Молоко $-x_4$	Картофель $-x_5$	Крупа гр. $-x_6$	1
Белки	-4	0	-10	-14	-33	-88	-300
Жиры	-7	-948	0	0	-35	-23	-300
Углеводы	-392	0	-74	-190	-44	-634	-1500
ккал	-1870	-8820	-350	-840	-640	-3170	-10000
Витамин A	0	0	-90	0	-0,5	0	-4,5
Витамин B ₁	-1,5	0	-0,6	-1	-0,5	-5	-6
Витамин B ₂	-1,3	0	-0,6	-0,5	-1,9	-2,4	-7,5
Витамин PP	-4,5	0	-4	-0,9	-1	-44	-45
Витамин C	0	0	-700	-100	-10	0	-210
Φ	-0,2	-1,62	-0,2	-0,1	-0,2	-0,2	-0,6

Таблица 8

**Программа преобразования симплекс-таблицы с числом
 $m \leq 11$ столбцов**

↑	F8	÷	/—/	P7	C/P	ПП	4	ПП	4	ПП	4
ПП	4	ПП	4	ПП	4	ПП	4	F3	ПП	F5	F4
ПП	F5	F5	P6	F2	←	P2	↑	F7	X	↑	F6
+	C/P	B/O									

После ввода этой программы в программную память заносят в регистр 8 разрешающий элемент a_{ks} преобразуемой таблицы, а остальные элементы a_{kj} разрешающей строки в порядке их следования – в регистры С1, С2, , С6, 2, 3, 4, 5. Затем вводят в регистр X элемент a_{is} разрешающего столбца в i -й неразрешающей строке и нажимают клавишу В/О и С/П. Записав высвечиваемое значение преобразованного элемента a'_{is} в новую таблицу, поочередно вводят в регистр X элементы a_{i1}, a_{i2}, \dots той же строки (пропустив a_{is}), нажимая после каждого ввода клавишу С/П и регистрируя в новой таблице высвечиваемые значения a'_{i1}, a'_{i2}, \dots . Если $m > 7$, то после преобразования элементов очередной неразрешающей строки следует "довернуть" стек памяти против часовой стрелки, чтобы элемент a_{1j} оказался в регистре С1. Выполнив подобным образом преобразование элементов всех неразрешающих строк, вводят -1 в регистр X и нажимают клавиши В/О и С/П, регистрируя преобразованное значение разрешающего элемента a_{ks} в новой таблице. Затем, очищая перед каждым пуском программы регистр X, нажимают требуемое число раз клавишу С/П, регистрируя после каждого выполнения программы очередное значение преобразованного элемента a'_{kj} разрешающей строки.

С помощью составленной программы после пяти преобразований исходной симплекс-таблицы успешно решаем задачу с результатом: хлеба 2,23 кг, молока 4,03 л, гречневой крупы 0,68 кг, растительного масла 0,135 г, морко-

ви 0,242 кг. При стоимости 1,93 руб. этот рацион обеспечивает по 300 г белков и жиров, 1500 г углеводов, 10175 ккал, 23,8 мг витамина А, 8,9 мг витамина В, 12,33 мг витамина В₂, 45 мг витамина РР и 210 мг витамина С.

Если в дневной рацион включить 0,9 кг картофеля, 0,5 кг яблок, 100 г сахара, 0,5 кг говядины, 100 г сливочного масла, 0,5 кг помидоров и 100 г сметаны, то при стоимости 2,03 руб. получаем 117 г белков, 150,6 г жиров, 339,4 г углеводов, 3277 ккал, 1,9 мг витамина В₁, 32,6 мг витамина РР, 1,75 витамина В₂ и полностью удовлетворяем потребность семейства в витаминах А и С.

Вычислив дефицит по каждому показателю и сохранив в симплекс-таблице предыдущий перечень продуктов, получим для них следующие оптимальные данные: хлеба 3,4 кг; масла растительного 0,107 кг; молока 0,7 л, что при полной стоимости суточного рациона в 3 руб. 3 коп. обеспечивает выполнение всех требований.

3.2. Ваш день рождения и биоритмы

Если спросить Вас, читатель, в какой день недели Вы родились, то вряд ли Вы сумеете ответить на этот вопрос. Однако ответ легко можно получить, используя микрокалькулятор. Для этого прежде всего необходимо научиться вычислять число дней между двумя датами Д₁. М₁. Г₁ и Д₂. М₂. Г₂, где Д, М и Г – день, месяц и год.

Интервал С между двумя датами разбиваем на три составляющих, вычисляемых по разностям дней, месяцев и лет в обеих датах. Первую составляющую вычисляем по формуле С = 365 (Г₂ – Г₁) + В, где Г₁ и Г₂ – годы в заданных датах, а В – число високосных лет между ними. Аналогично выглядит и вторая составляющая С₂ = 30 × X (М₂ – М₁) + П, где М₂ и М₁ – номера месяцев в датах, а П – поправка, зависящая от числа дней в месяце. Вычислив С₃ = Д₂ – Д₁ и сложив все составляющие, получаем

интересующий нас ответ. Однако при попытке практического применения этого алгоритма возникают непредвиденные трудности. Поправка B , как оказывается, зависит не только от чисел Γ_2 и Γ_1 , но и от чисел M_2 и M_1 : если одна из дат приходится на високосный год, то для даты до и после 28 февраля поправка оказывается различной.

Выход из положения находится, если задуматься над происхождением названий последних месяцев года (*septem* – семь, *octo* – восемь, *nono* – девять и *deca* – десять). В этом случае "перенесся" встречу Нового года на 1 марта, используем подстановку $M' = M + 9$, $\Gamma' = \Gamma - 1$, если $M < 3$, и $M' = M - 3$, $\Gamma' = \Gamma$ при $M \geq 3$. Тогда с 1 марта до начала месяца M приходится дней: до $M' = 1$ (до 1 апреля) – 31, до $M' = 2$ (до 1 мая) – 61, до $M = 3$ (до 1 июня) – 92, до $M' = 4$ (до 1 июля) – 122, до $M' = 5$ (до 1 августа) – 153, до $M' = 6$ (до 1 сентября) – 184, до $M = 7$ (до 1 октября) – 214, до $M' = 8$ (до 1 ноября) – 245, до $M' = 9$ (до 1 декабря) – 275, до $M' = 10$ (до 1 января) – 306, до $M' = 11$ (до 1 февраля) – 337.

В "смещенном" календаре, например, дата 12.12.1980 превращается в 12.9.1980, а дате 21.2.1981 соответствует дата 21.11.1980. В этом случае поправку для месяцев можно будет вычислить по формуле $\Pi = E(30,6 M' + 0,4)$, где $E(\)$ означает целую часть числа в скобках. Эта формула дает требуемые результаты $\Pi(1) = E(31) = 31$, $\Pi(2) = E(61,6) = 61$, $\Pi(3) = E(92,2) = 92$ и т. д.

Учтя подобным образом и число високосных лет в заданном интервале, сводим вычисление интервала в днях между двумя заданными датами к выполнению следующих несложных операций.

1. Вычислить $M' = M + 9$, $\Gamma' = \Gamma - 1$, если $M < 3$; $M' = M - 3$, $\Gamma' = \Gamma$, если $M \geq 3$.

2. Вычислить искомое число дней по формуле $C = \Delta_2 - \Delta_1 + E(30,6 M'_2 + 0,4) - E(30,6 M'_1 + 0,4) + E(1461 \Gamma'_2 / 4) - E(1461 \Gamma'_1 / 4)$.

Эти операции легко выполнить на микрокалькуляторе любого типа, но для дат, связанных с историей нашей страны, этот алгоритм справедлив лишь с 14 февраля 1918 г. Со времен Петра Первого время в России исчислялось по юлианскому календарю. Согласно этому летоисчислению ("старому стилю") каждый четвертый год имел в феврале 28 дней и назывался високосным. Однако средний юлианский год отличается от астрономического (365,2422 дней) на 0,0078 дня и эта маленькая погрешность вызывала постепенное отставание "старого стиля" от времен года.

В 1582 г. папа римский Григорий ХХIII реформировал календарь в странах с католическим вероисповеданием: было опущено 10 дней и пятницу 5 октября стали считать пятницей 15 октября. Чтобы в дальнейшем ошибка была меньшей, реформа ("новый стиль") не относила к високосным годы столетий 1700, 1800, 1900, 2100 и последующие, в которых число веков не делится на четыре. Эта мера уменьшила ошибку примерно до одного дня за 3000 лет.

В англоязычных странах, где преобладали протестанты, григорианский календарь был принят в 1752 г., а в нашей стране – с 14 февраля 1918 г. К этому времени расхождение старого и нового стилей достигло 13 дней и поэтому 1 февраля 1918 г. по старому стилю стали считать 14 февраля 1918 г. по новому стилю. Отсюда вытекают выводы.

1. При определении числа дней между датами в истории нашей страны, разделенных датой перехода на новый стиль, из результата, полученного по приведенной ранее формуле, следует вычесть 13 дней.

2. Если Вы определяете даты после 2100-го года, то годы 2100, 2200, 2300, 2500 и т. п. не являются високосными.

3. Если Вас интересует зарубежная история, то необходимо учесть время перехода в этой стране на григорианский календарь.

Если вернуться к первоначальной задаче, то для определения дня недели по заданной дате достаточно вычислить остаток от деления на семь числа дней между этой датой и датой, приходящейся на воскресенье. Полученное число можно считать кодом дня недели: 0 – воскресенье, 1 – понедельник, 2 – вторник, 3 – среда, 4 – четверг, 5 – пятница, 6 – суббота.

Очевидно, нет необходимости вычислять полное число дней между такими датами, а достаточно учесть, что в каждый невисокосный год начало недели сдвигается на один день, а в високосный – на два. Из каждого 30 дней можно исключить полных четыре недели, учтя только двухдневный сдвиг. Эти соображения позволяют составить следующую формулу для определения дня недели по заданной дате Д.М.Г.:

$K = (D + E(2,6M' + 0,4) + E(5G'/4) + X) - 7 \cdot E((D + E(2,6M' + 0,4) + E(5G'/4) + X)/7)$, где M' и G' определяют по заданному месяцу и году, как и ранее, а X – целое число, которое подбирают так, чтобы получить совпадение дня недели для определенной даты. Так, определив по календарю, что 8 марта 1981 г. приходится на воскресенье, вычислим $M' = 0$, $G' = G = 1981$ и $K = (8 + E(0,4) + E(2476,25) + X)/7$. Результат делится на 7 без остатка ($K = 0$), если принять $X = 1$. Это значение X позволяет определить дни недели для любой даты между 14.2.1918 и 28.2.2100. Для дат, предшествующих введению нового стиля в нашей стране, следует принять $X = 0$, так как пропуск 13 дней смещает начало недели на один день.

Определим теперь, какому дню недели соответствует, например, 14 января 1971 г. Вычисляя $M' = 2 + 8 = 10$, $G' = 1971 - 1 = 1970$ и $K = (14 + E(26 + 0,4) + E(2462,5 + 1) - 7E((14 + E(26 + 0,4) + E(2462,5) + 1)/7)) = 2503 - 2499 = 4$, получаем, что это соответствует четвергу.

Полученное аналитическое выражение позволяет составить программу автоматического вычисления дня неде-

ли с помощью микрокалькулятора "Электроника Б3-21" (Д = Р2, М = Р3, Г = Р4, 1918 = Р8, FX = К).

Программа определения дня недели по заданной дате

F3	3	-	x<0	P2	F4	1	-	P4	F3	9	+
1	3	X	2	+	5	ПП	PC _x	F4	↑	F8	-
x≥0	F,	1	ПП	7	F4	5	X	4	ПП	PC _x	P7
7	ПП	PC	F7	↑	F6	-	C/П	÷	1	ВП	7
XY	+	XY	-	7	x	P6	F2	+	P2	B/O	

Для первых выпусков микрокалькуляторов этого типа следует использовать следующую программу при тех же исходных данных:

F3	3	-	x<0	P2	F4	1	-	P4	F3	9	+
1	3	X	5	ПП	FC _x	F4	↑	F8	-	5	X
x<0	F,	4	-	1	7	+	4	ПП	FC _x	7	/-
ПП	FC	↑	←	+ x<0	7	7	+	C/П	÷	1	
ВП	7	XY	-	-	6	X	→	F2	+	P2	B/O

Эти программы (пуск клавишами В/О и С/П) определяют дни недели для любых дат отечественной истории со времени принятия юлианского календаря Петром Первым до 28 февраля 2100 г., за исключением дат в феврале 1918 г. В последнем случае вычисляемый программой ход увеличивают на единицу. Так, для 9 мая 1945 г. по этим программам (9 = Р2, 5 = Р3, 1945 = Р4, 1918 = Р8) получим FX=3, что соответствует среде, а для 9 января 1905 г. (9 = Р2, 1 = Р3, 1905 = Р4, 1918 = Р8) получим FX = 0 – это день "кровавого воскресенья" в нашей истории.

Определяя дни недели, сталкиваемся с вопросом делимости числа на заданный делитель. Математики относят этот вопрос к теории сравнения чисел с основным соотношением $a \equiv b \pmod{N}$, которое читается как "число a сравнимо с числом b по модулю N " и означает, что разность чисел a и b делится на N без остатка. Используя это

выражение, последнюю из формул можно записать в виде $K = (D + E(2,6M' + 0,4) + E(5G'/4) + X) \pmod{7}$, где X равно 0 или 1 для нового или старого стилей. Это не единственное полезное приложение теории сравнений. Ее, в частности, можно использовать и для составления расписания соревнований m команд в $(m-1)$ -круговом турнире.

Обозначим буквами x номер команды ($x \leq m-1$), для которой составляется расписание по турнам, r – номер тура, y_r – номер команды, с которой встречается команда x в r -м туре. При четном числе m (если m нечетно, то его увеличивают на единицу, добавив m -ю фиктивную команду, встреча с которой означает, что команда x свободна), если $y_r = x$, то $y_r = m$, если $r - x > 0$, то $y_r = r - x$, и при $r - x \leq x$ команда x встречается с командой $y_r = r - x + (m-1)$. Составление расписания игр можно автоматизировать при помощи следующей программы.

Программа составления расписания игр в турнире

P3	F2	1	P4	↑	F3	XY	–	$x \geq 0$	2	1	+
↑	F2	–	/–	↑	F3	–	$x=0$	÷	F2	↑	XY
C/P	F4	1	+	БП	0						

После ввода программы в регистр 2 заносят четное число m команд (если оно нечетно, то добавляют единицу), в регистр X – номер $x \leq m-1$ команды, для которой составляется расписание тура, и нажимают клавиши В/О и С/П. Записав высыпавшийся номер y_1 команды, с которым команда x встречается в первом туре, нажимают $m-2$ раз клавишу С/П, записывая после каждого выполнения программы номера y_r команд, с которыми встречается команда x . Повторив вычисления для всех команд до $m-1$, при четном m для команды $x = m$ номера команд y_r записывают как номер, отсутствующий в предыдущих номерах столбца.

Так, для составления расписания встреч шести команд исходные данные $b = P_2$, $1 = P_X$ позволяют вычислить элементы первой строки таблицы расписания (табл. 9). Изменив содержимое регистра X на 2, 3, 4 и 5, вычисляют остальные элементы таблицы, кроме элементов шестой строки, которыми являются номера команд, отсутствующие в предыдущих столбцах.

Таблица 9

Расписание встреч для шести команд

x	r					
	1	2	3	4	5	6
1	5	6	2	3	4	
2	4	5	1	6	3	
3	6	4	5	1	2	
4	2	3	6	5	1	
5	1	2	3	4	5	
6	3	1	4	2	5	

Опыт подобных вычислений поможет читателям удовлетворить интерес к модной одно время теории биоритмов. Согласно этой теории, состояние человека со дня его рождения периодически изменяется, и дни подъема физических и душевных сил сменяются днями их упадка. При этом физическое состояние изменяется с периодом в 23 дня, эмоциональное — 28 дней и интеллектуальное — 33 дня (рис. 25). Первая половина этих периодов соответствует подъему, а вторая спаду, причем неустойчивым состоянием соответствуют критические дни перехода от подъема к спаду и наоборот. Наихудшее состояние приходится на день перехода всех состояний через нуль ("тройной нуль"), затем следуют "двойные нули", "одиночные нули" при отрицательной фазе других состояний и отрицательной фазе всех состояний.

По графикам биоритмов (рис. 25) нетрудно определить, что уровень каждого из них определяется формулой $K_i = \sin(2\pi D/T_i)$, где T – период i -го биоритма, а D – число дней, прошедших со дня рождения.

При использовании арифметического микрокалькулятора достаточно определить дробную часть от D/T_i и сравнить соответственно с числами 0,4782; 0,5 и 0,4848. Если результат равен нулю или одному из этих чисел соответственно для физического, эмоционального и интеллектуального состояний, то Вы переживаете критический день. Если дробная часть меньше соответствующего числа, то фаза биоритма положительна, иначе – отрицательна. Например, для $D = 10258$ получим критический день для физического состояния ($10258/23 = 446$), положительную фазу для эмоционального ($10258/28 = 366,35714$ и $0,357 < 0,5$) и отрицательную – для интеллектуального ($10258/33 = 310,84848$ и $0,84848 > 0,4848$).

При использовании программируемого микрокалькулятора "Электроника Б3-21" следующая программа при $D = RX$ вычисляет после нажатия клавиши В/О и С/П трехзначное число, содержащее все три оценки физического, эмоционального и интеллектуального состояний (1 –

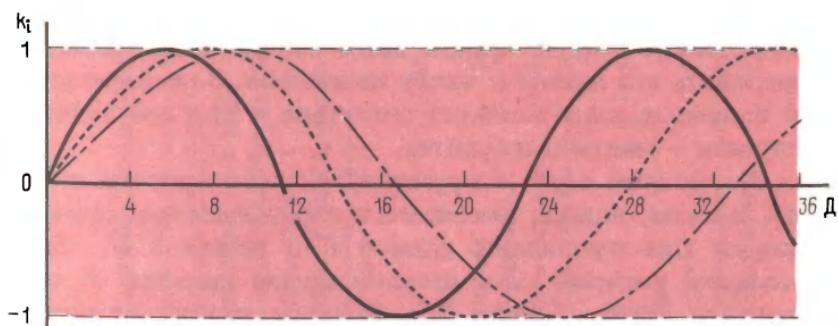


Рис. 25. Графики биоритмов

критический день, 2 – отрицательная фаза, 5 – положительная фаза).

Программа вычисления оценок биоритмов

P2	2	3	ПП	,	0	X	P7	2	8	ПП	,
X	↑	F7	+	P7	3	3	ПП	,	F7	+	C/П
F2	1	+	БП	РО	P3	F2	↑	F3	÷	2	P8
X	↑	π	X	sin	2	0	1/x	+	$x \geq 0$	F9	5
P8	$1/x$	$-x < 0$	F9	1	P8	F8	1	0	B/O		

При последующих нажатиях клавиши С/П эта программа увеличивает Д на единицу и вычисляет оценки биоритмов для следующих дней.

Для "убедительности" подобных прогнозов были оформлены результаты вычислений по этой программе на "официальном бланке", подобном приведенному в табл. 10.

3.3. Покупать ли шкаф?

С задачей перемещения мебели через повороты коридоров и дверные проемы не раз сталкивается большинство читателей. Сложность ее решения связана не с размерами современных квартир, а с размерами мебели, ибо Козьма Прутков мудро заметил, что "нет столь великой вещи, которую не преъзошла бы вещь, еще более великая". Поэтому перед покупкой не только антикварного, но и обычного шкафа следует прежде всего определить, удастся ли доставить его целым к месту назначения. В решении этой и подобных задач поможет геометрия и наш постоянный спутник – микрокалькулятор.

Рассмотрим перенос крупногабаритного предмета в виде, скажем, ящика, имеющего в горизонтальном сечении форму прямоугольника длиной L и шириной H , через коридор шириной C и дверной проем шириной D при толщине стенок T (рис. 26). Заметим, что последующий анализ применим и к задаче протаскивания мебели через поворот коридора (показанный штриховой линией на

Таблица 10

Биоритмы Ивана Ивановича Иванова

Дата рождения 25.8.1950

Дата составления 15.5.1982

Число II положительных пят 11 586

Оценки состояния

Положительная фаза 5, отрицательная фаза 2, критический день 1.
Однако состояния.

Число	Состояние:	Май												Июнь
		15	16	17	18	19	20	21	22	23	24	25	26	
Физическое	Состояние:	2	2	2	2	2	2	1	5	5	5	5	5	5
Эмоциональное	Состояние:	2	2	2	2	2	2	1	5	5	5	5	5	5
Интеллектуальное	Состояние:	5	5	5	5	5	5	5	5	5	5	1	2	2

Окончание табл. 10

Число	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Состояние:														
физическое	5	5	5	5	5	5	5	1	2	2	2	2	2	2
эмоциональное	2	1	5	5	5	5	5	5	5	5	5	5	5	5
интеллектуальное	5	5	5	5	5	5	5	5	5	5	5	5	5	1

Июль

Число	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Состояние:																
физическое	2	2	2	2	2	1	5	5	5	5	5	5	5	5	5	5
эмоциональное	5	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
интеллектуальное	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

рис. 26, а) или проем с очень толстыми стенками, для чего достаточно принять $T \geq L - C$.

При решении этой задачи естественно предположить, что выполняются условия $L > H$ и $L > C$, так как в противном случае при $D > H$ достаточно внести ящик, как показано на рис. 26, а. Более того, если даже ящик внесен в коридор "боком", то при $D > H$ его удастся развернуть в нужное положение. При исходных условиях наш ящик начнет проходить через дверной проем, если их углы удастся совместить, что соответствует условию $D > D_0$ (рис. 26, б). А так как $D_0 = H / \sin \alpha_0$ и $\sin \alpha_0 = C / L$, то это неравенство принимает вид $D \geq HL / C$.

Однако при выполнении этого условия радоваться еще рано. Может оказаться, что при дальнейшем продвижении ящика он застрянет. Это случится, если ширина дверного

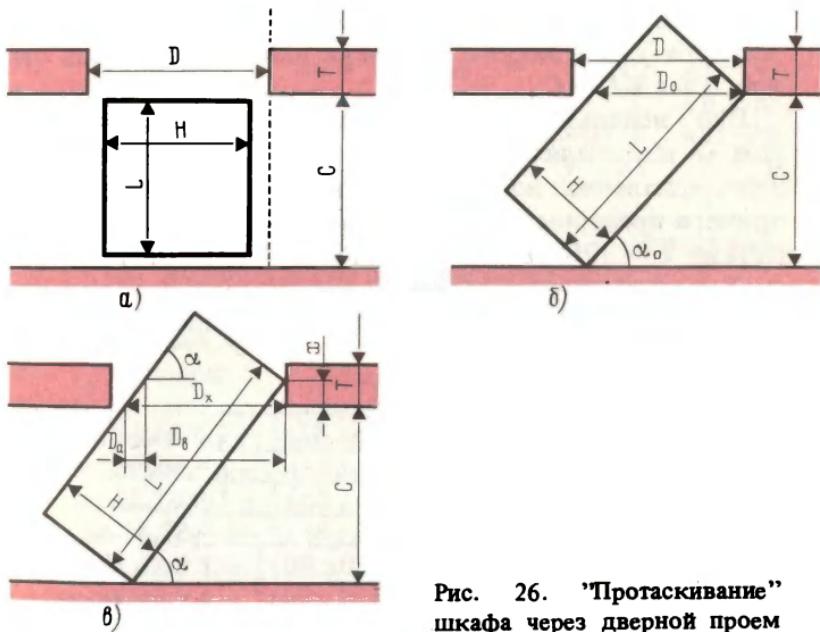


Рис. 26. "Протаскивание" шкафа через дверной проем

проема D меньше требуемой для прохождения ящика (рис. 26, в) :

$$D_x = D_a + D_b = \frac{x \cos \alpha}{\sin \alpha} + \frac{H}{\sin \alpha} = \frac{x \sqrt{1 - \sin^2 \alpha}}{\sin \alpha} + \frac{H}{\sin \alpha},$$

где x – расстояние "переднего" угла ящика от стенки коридора. Подставив $\sin \alpha = (C + x)/L$, получим

$$D_x = \frac{x \sqrt{1 - (C + x)^2/L^2} + H}{(C + x)/L} = \frac{HL}{C+x} + x \sqrt{\left(\frac{L}{C+x}\right)^2 - 1}.$$

Таким образом, условие прохождения ящика опишется неравенством $D \geq D_{x \max}$, где $D_{x \max}$ – максимальное значение D_x . Хотя мы умеем находить максимальные значения функций сложными методами, наиболее простое решение заключается в вычислении ряда значений D_x до прохождения максимума при изменении x с достаточно малым шагом Δx в интервале $0 \leq x \leq T$ или, если задача решается для поворота коридора или очень толстых стеконок, $0 \leq x \leq L - C$.

При использовании микрокалькулятора с оператором \sqrt извлечения корня значения D_x можно вычислять непосредственно по приведенной формуле. В качестве примера проверим, удастся ли нам протащить шкаф с размерами $L = 120$ см и $H = 44$ см через проем шириной $D = 68$ см при ширине коридора $C = 80$ см и толщине стеконок $T = 20$ см. Приняв $\Delta x = 2$ см, последовательно вычисляем:

$$D_0 = 44 \times 120 : 80 = 66 \text{ см};$$

$$D_1 = 44 \times 120 : 82 + 2 \cdot \sqrt{(120 : 82)^2 - 1} = 66,53 \text{ см};$$

$$D_2 = 44 \times 120 : 84 + 4 \cdot \sqrt{(120 : 84)^2 - 1} = 66,94 \text{ см};$$

$$D_3 = 44 \times 120 : 86 + 6 \cdot \sqrt{(120 : 86)^2 - 1} = 67,23 \text{ см};$$

$$D_4 = 44 \times 120 : 88 + 8 \cdot \sqrt{(120 : 88)^2 - 1} = 67,42 \text{ см};$$

$$D_5 = 44 \times 120 : 90 + 10 \cdot \sqrt{(120 : 90)^2 - 1} = 67,49 \text{ см};$$

$$D_6 = 44 \times 120 : 92 + 12 \cdot \sqrt{(120 : 92)^2 - 1} = 67,44 \text{ см}.$$

Итак, $D_{x \text{ max}} \approx 67,49$ см < 68 см и, следовательно, можно платить за шкаф в кассу мебельного магазина.

Для решения задачи на арифметическом микрокалькуляторе без оператора $\sqrt{\cdot}$ целесообразно преобразовать условие $D \geq D_{x \text{ max}}$ к виду

$$(D - \frac{HL}{C+x})^2 \geq x^2 \left(\left(\frac{L}{C+x} \right)^2 - 1 \right)$$

и проверить его выполнение в том же интервале изменения x .

Владельцы микрокалькулятора "Электроника Б3-21" и ряда других могут воспользоваться следующей программой автоматических вычислений при исходных данных L , H , C и T , выраженных в сантиметрах, занесенных соответственно в регистры 2, 3, 4 и 5.

Программа "протаскивания ящика"

F2	0	P6	F3	X	↑	F4	÷	P8	F6	2	+
P6	↑	F5	—	$x < 0$	P8	F4	+	↑	F2	XY	÷
↑	x^2	→	F3	X	←	1	—	$\sqrt{\cdot}$	↑	F6	X
↑	→	+	↑	F8	XY	P8	—	$x \geq 0$	1	XY	P8
F8	9	1/x	+	1	BП	6	XY	+	XY	—	C/П

После выполнения программы (пуск — клавишами В/0 и С/П) на индикаторе высвечивается значение $D_{x \text{ max}}$, округленное до одной цифры после запятой (неокругленное значение хранится в регистре 8). В программе принято приращение $\Delta x = 2$ см, но его можно изменить заменой оператора 2 по адресу 15. В нашем примере не учитывалась возможность наклона ящика. Высота l двери и высота h ящика должны быть меньшими размера L , так как иначе ящик можно поставить "на попа" и внести в дверь. Но при $L > l$ и $L > h$ может оказаться возможным так наклонить ящик (рис. 27, а), чтобы он прошел в дверь и при $D < D_{x \text{ max}}$. Для этого нужно, чтобы L оказалась

большой проекции ящика на горизонтальную плоскость (рис. 27, б) :

$$L' = \frac{2Ll h + (L^2 - h^2) \sqrt{L^2 + h^2 - l^2}}{L^2 + h^2}.$$

Например, при $L = 2$ м, $h = 0,5$ м и $l = 1,8$ м по этой формуле $L' = 1,733812 < L$. Поэтому в случае неудачи в протаскивании ящика можно попытаться проверить возможность наклона ящика, подставив в ваши формулы L' вместо L . Нам остается лишь подчеркнуть еще раз, что подобная проверка нужна лишь при $h < L$ и $l < L$.

Задачи по геометрии, с одной из которых мы только что столкнулись, обычно связаны с вычислением элементов треугольников. Чтобы такой треугольник не оказался для Вас "роковым", целесообразно запастись алгоритмами и программами решения основных типов подобных задач.

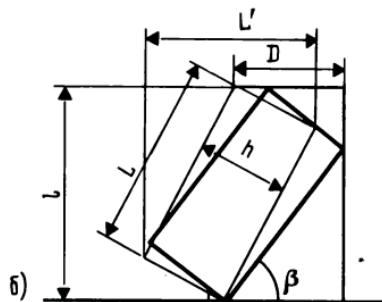
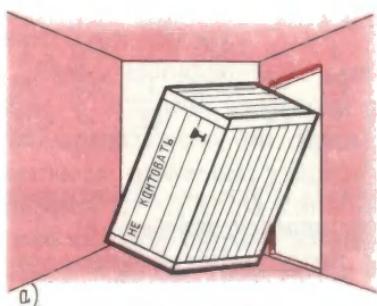


Рис. 27. "Протаскивание" шкафа с наклоном

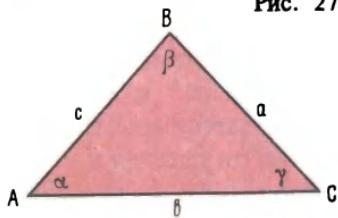


Рис. 28. Элементы треугольника

Шесть элементов треугольника — стороны a , b , c и углы α , β , γ (рис. 28) — связаны между собой формулой $\alpha + \beta + \gamma = \pi$ или $\alpha + \beta + \gamma = 180^\circ$, если углы измеряются в градусах; соотношениями теоремы синусов

$$a/\sin \alpha = b/\sin \beta = c/\sin \gamma$$

и теоремы косинусов

$$a^2 + b^2 - 2ab \cos \gamma = c^2; a^2 + c^2 - 2ac \cos \beta = b^2; b^2 + c^2 - 2bc \cos \alpha = a^2.$$

Кроме того, в расчет можно включить определение площади треугольника

$$S = \frac{ab \sin \gamma}{2} = \frac{ac \sin \beta}{2} = \frac{bc \sin \alpha}{2} = \sqrt{p(p-a)(p-b)(p-c)},$$

где $p = (a + b + c)/2$ — полупериметр, а также радиуса $r = 2S/(a + b + c)$ вписанной и радиуса $R = abc/4S$ описанной окружностей.

Рассмотрим решения нескольких основных задач подобного типа.

1. *Определение стороны c и углов α и β треугольника по двум сторонам a и b и углу γ между ними.*

При выборе подходящих для решения этой задачи расчетных выражений необходимо правильно определить углы, так как несовпадение интервала $[0, \pi]$ их возможных значений, определяемых с помощью обратных тригонометрических функций (как главное значение), может усложнить вычисление. Для рассматриваемой задачи в этом отношении наиболее удобны формулы

$$\alpha = \pi/2 + \operatorname{arctg} ((a \cos \gamma - b)/a \sin \gamma); c = a \sin \gamma / \sin \alpha; \\ \beta = \pi - \alpha - \gamma.$$

При использовании микрокалькулятора "Электроника Б3-21" приходится обеспечить вычисление функции $\operatorname{arctg} x$. Для этого можно воспользоваться следующей программой,

обеспечивающей большую точность результата по сравнению с ранее приведенной расчетной формулой для $\arctg x$.

Программа вычисления стороны c , углов α и β и площади S треугольника по сторонам a и b и углу γ между ними

F4	↑	F5	X	e^{jx}	-	F2	XY	X	P8	←	X
↑	F3	P2	-	↑	F8	÷	P3	x^2	↑	F6	+
↑	F7	X	✓	+	4	$1/x$	+	✓	↑	F3	XY
÷	9	0	XY	X	+	P4	C/П	↑	F5	X	e^{jx}
F8	XY	÷	P3	C/П	F2	X	2	÷	C/П	БП	РО

Исходные данные a , b и γ (в градусах) заносят соответственно в регистры 2, 3, 4, а числа $\pi/180$; 0,96573 и 1,62 заносят соответственно в регистры 5, 6 и 7. После пуска программы нажатием клавиш В/О и С/П и ее выполнения высвечивается значение угла, два последующих нажатия клавиши С/П приводят к вычислению стороны c и площади треугольника S . Снова нажав клавиши В/О и С/П, получают значение угла β , а после двух следующих пусков программы нажатием клавиши С/П получают сторону a и снова площадь S . Цикл вычислений можно повторять для контроля точности вычислений и оценки операционных погрешностей. Так, при $a = 5$, $b = 6$ и $\gamma = 130,5415^\circ$ последовательно получим $\alpha = 22,33204$; $c = 9,999847$; $S = 11,39903$; $\beta = 27,12916$; $a = 4,999673$; $S = 11,39903$; $\gamma = 130,5390$; $b = 6,000174$; ...

Если нет необходимости вычислять S , то вместо соответствующих операторов в программу можно внести переход от градусов к радианам и операторы БП, РО автоматического возврата в начало программы.

2. *Определение угла α и сторон b и c по стороне a и углам β и γ .*

В этом случае достаточно выполнить расчет по формулам: $\alpha = 360^\circ - \beta - \gamma$; $b = a \sin \beta / \sin \alpha$; $c = a \sin \gamma / \sin \alpha$. Владельцы микрокалькулятора "Электроника Б3-21" и

ему подобных могут воспользоваться следующей программой ($a = P2$, $\beta = P3$, $\gamma = P4$), приводимой ниже.

Программа вычисления угла и двух сторон треугольника

F2	1/x	P8	F3	↑	F4	+	1	8	0	XY	-
P5	C/П	ПП	P7	1/x	P8	F3	ПП	P7	P6	C/П	F4
ПП	P7	P7	C/П	↑	F8	÷	2	÷	↑	F2	X
↑	F6	X	C/П	BП	PO	1	8	0	÷	↑	π
X	sin	↑	F8	X	B/O						

После четырех выполнений программы (первый пуск – клавишами B/O и C/П, последующие – C/П) последовательно получим α , b , c и S , причем высвечиваемые значения α , b и c автоматически записываются также в регистры 5, 6 и 7 соответственно.

3. Определение углов треугольника по его сторонам.

В этом случае проще всего воспользоваться соотношениями:

$$\alpha = \arccos \frac{a^2 - b^2 - c^2}{2bc} ; \quad \beta = \arccos \frac{b^2 - a^2 - c^2}{2ac} ;$$

$$\gamma = \arccos \frac{c^2 - a^2 - b^2}{2ab}$$

или следующей программой автоматических вычислений ($a = P2$, $b = P3$; $c = P4$).

Программа вычисления углов треугольника по его сторонам

F2	P5	x^2	↑	F3	P2	x^2	+	↑	F4	P3	x^2
XY	-	↑	F5	P4	÷	↑	F2	÷	2	÷	P8
x^2	1	XY	-	5	÷	8	,	1	X	\sqrt	+
1	+	\sqrt	↑	F8	XY	÷	9	0	XY	X	+
C/П	BП	PO									

После каждого нажатия клавиши C/П (первый раз – клавиш B/O и C/П) последовательно высчитываются углы

$\gamma, \alpha, \beta, \gamma, \dots$ в градусах. Площадь треугольника можно дополнительно вычислить в обычном режиме вводом предложения: $\uparrow \pi \times 180 \div \sin \uparrow F2 \times \uparrow F4 \times 2 \div$.

4. Определение элементов треугольника по двум сторонам и углу, прилегающему к одной из них (например, a, b и α).

Задача имеет два решения, если $b \sin \alpha < a < b$, одно, если $a > b$, и ни одного, если $a < b \sin \alpha$. Вычисления следует начать с определения $\beta_1 = \arcsin(b \sin \alpha / a)$; $\beta_2 = \pi - \beta_1$. Тогда $\gamma_{1,2} = 180 - \alpha - \beta_{1,2}$ и $c_{1,2} = b \sin \gamma_{1,2} / \sin \beta_{1,2} = a \sin \gamma_{1,2} / \sin \alpha$. Если γ или c отрицательно, то соответствующее решение отбрасывается. Вычисления можно автоматизировать с помощью следующей программы ($a = P2$, $b = P3$, $\alpha = P4$, $\pi/180 = P5$).

Программа вычисления элементов треугольника по двум сторонам и углу

F4	\uparrow	F5	\times	\sin	\uparrow	F2	\div	P6	\uparrow	F3	x
P7	x^2	1	XY	-	5	\div	8	,	1	X	\sqrt
+	1	+	\sqrt	\uparrow	F7	XY	\div	9	0	X	C/P
1	8	0	XY	-	P8	\uparrow	F4	-	C/P	\uparrow	F5
X	\sin	\uparrow	F6	\div	C/P	F8	BП	/-			

После каждого выполнения программы на индикатор последовательно выводятся значения $\beta_1, \gamma_1, c_1, \beta_2, \gamma_2, c_2, \dots$.

Учитывая большие возможности микрокалькулятора, была разработана программа для подготовки к экзамену по географии. Основой программы является несколько "экзаменационных" билетов, каждый из которых содержал по шесть вопросов, например:

1. Сколько веков существует город Киев?
2. В какое море впадает Волга? (1. Азовское 2. Черное
3. Каспийское)
3. В каком году Колумб открыл Америку?

4. Что такое Лимпопо? (1. Индейское племя. 2. Озеро в Боливии 3. Река в Африке. 4. Австралийское животное)
 5. Какая река самая длинная? (1. Амазонка 2. Волга 3. Миссисипи 4. Нил)
 6. Сколько рек вытекает из озера Байкал?

На каждый вопрос такого билета нужно ответить числом или, в частности, порядковым номером имеющегося в билете ответа. Проверяет правильность ответов и выставляет оценку экзаменующемуся микрокалькулятор "Электроника Б3-21" со следующей программой.

Программа-экзаменатор

Cx	P4	1	P5	P5	C/П	↑	F3	←	P3	—	x=0
X	F4	1	+	P4	F5	1	+	P5	7	—	x=0
F↑	F3	←	F4	1	—	P6	2	—	x<0	F6	2
P6	F6	0	0	C/П							

После ввода этой программы в программную память в регистры C1, ..., C6 стека памяти следует соответственно ввести правильные ответы на вопросы билета в порядке из очередности и нажать клавиши В/О и С/П, что приведет к высвечиванию номера вопроса, на который должен ответить экзаменующийся. Он вводит в регистр X свой ответ на вопрос и нажимает только клавишу С/П, что приводит к высвечиванию номера следующего вопроса. После ответа на шестой вопрос и нажатия клавиши С/П высвечаются цифры 00 ("Опрос окончен"), после чего при нажатии клавиши XY индицируется оценка в баллах 5, 4, 3 или (если правильных ответов меньше четырех) 2.

3.4. "Регата" на клавишах

По просьбе друзей мы попытались имитировать с помощью микрокалькулятора "Электроника Б3-21" работу игральных автоматов. Например, артиллерийскую стрельбу на игровых автоматах несложно промоделировать, учитывая, что снаряд, вылетающий из орудия (рис. 29)

с начальной скоростью v под углом возвышения θ , падает (без учета сопротивления воздуха) на расстоянии $d = v^2 \sin 2\theta/g$, где $g \approx 9,8 \text{ м/с}^2$. Эту формулу удобно записать как $d = d_{\max} \sin 2\theta$, где $d_{\max} = v^2/g$ — наибольшее для орудия с начальной скоростью снаряда v (при $\theta = 45^\circ$) расстояние до места падения снаряда.

Задавшись d_{\max} , с помощью этой формулы легко вычислить разность между расстоянием d до места падения снаряда и расстоянием d_u до цели. Считая цель пораженной, если эта разность меньше радиуса Δr разрыва снаряда, несложно имитировать стрельбу с помощью следующей программы ($d_u = P7, 0 = P8, d_{\max} = P4, (\Delta r)^2 = P5$).

Программа игры "Навесная стрельба"

\uparrow	π	\times	9	0	\div	e^{jx}	F4	\times	\uparrow	F7	$-$
P2	F8	1	+	P8	F2	x^2	\uparrow	F5	$-$	$x < 0$	4
0	0	C/P	F2	C/P	БП	РО					

Перед каждым "выстрелом" в регистр X вводят угол θ возвышения орудия в градусах и нажимают клавишу С/П (первый раз В/О и С/П). После выполнения программы на индикаторе высвечивается расстояние от цели до места падения снаряда (при недолете — с отрицательным знаком) или, при поражении цели, цифры 00. В регистре 8

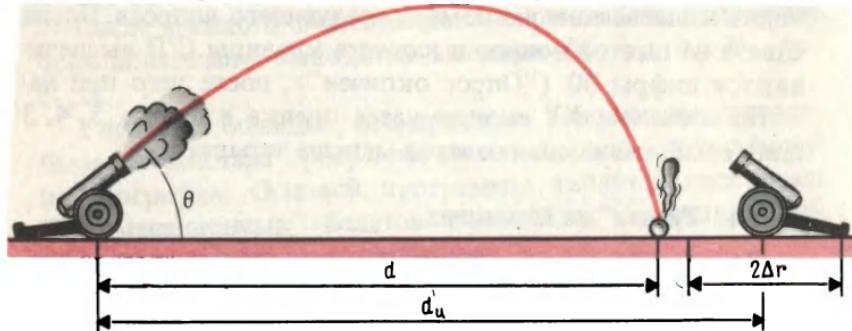


Рис. 29. Навесная стрельба

накапливается число выстрелов. Эта программа пригодна и для моделирования "артиллерийской дуэли" двух игроков, по очереди вводящих угол θ в регистр X и пускающих программу. Не следует лишь показывать противнику значение вводимого угла.

Положение цели в подобных играх можно задавать и двумя координатами – расстоянием $d_{ц}$ и углом $\varphi_{ц}$ между направлением на цель и некоторым начальным направлением. Выбор исходного положения цели целесообразно поручить микрокалькулятору, приняв в качестве исходных числа H часов и M минут начала игры и, например, формулы $\varphi_{ц} = (H + M) : 6M$; $d_{ц} = (\varphi_{ц} + 1) 10^4$. Микрокалькулятор выполняет и функции прибора, вычисляющего расстояние от цели до места падения снаряда

$$r = \sqrt{d^2 + d_{ц}^2 - 2dd_{ц} \cos \varphi},$$

где d – расстояние до места падения снаряда, а φ – угол между направлениями на цель и местом падения снаряда (рис. 30).

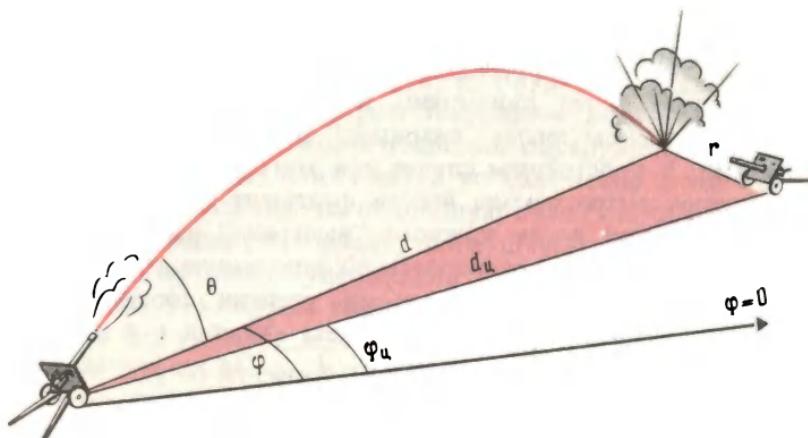


Рис. 30. Артиллерийский бой
10*

Программа игры "Артиллерийский бой"

+	XY	÷	6	÷	P2	1	+	1	ВП	4	X
P3	C _x	P8	C/П	P7	F2	—	cos	↑	F3	X	↑
F7	X	2	X	/—/	↑	F3	x ²	+	↑	F7	x ²
+	√	P5	F8	1	+	P8	F5	↑	F4	—	x<0
—	0	0	C/П	F5	БП	2					

Перед началом игры в регистр 4 вводят выбранное значение Δr , в регистры Y и X – соответственно числа H часов и M минут и нажимают клавиши В/О и С/П. После выполнения программы и высвечивания нуля открывают пристрелочную стрельбу, вводя угол $\theta \leq 1,57$ (в радианах) и d (в метрах) в регистры Y и X и нажимая только клавишу С/П. После каждого выполнения программы на индикаторе высвечивается расстояние от цели до места падения снаряда или, при попадании, цифры 00. В регистре 8 накапливается сумма выпущенных снарядов.

Эта программа пригодна и для двух участников, по очереди "обстреливающих" друг друга. Заметим, что если для первой из игр поиск выигрыша аналогичен поиску корня уравнения, то во второй игре он соответствует поиску минимума функции двух переменных.

Гораздо более динамичны и не беспроигрышны при неограниченном числе "снарядов" игры с движущимися целями. В простейшем случае для этого достаточно в рассмотренные программы ввести фрагмент $F7 \uparrow F6 - P7$, уменьшающий после каждого "выстрела" расстояние до цели на величину, предварительно занесенную в регистр 6. Реальным условием в большей степени соответствует модель, учитывающая время полета снаряда $t = a \sin \theta$, где $a = 2v/g = 2\sqrt{d_{\max}/g} \approx 0,6\sqrt{d_{\max}}$. За это время цель переместится на расстояние $t v_{ц}$, где $v_{ц}$ – скорость движения цели, и после каждого выстрела приблизится на расстояние $d_{i+1} = d_i - a v_{ц} \sin \theta$. Задача артиллеристов в этом случае существенно усложняется.

Используя приведенные формулы, составим следующую программу с исходными данными $d_{\max} = P4$, $(\Delta r) = P5$, $d_{ц} = P7$, $v_{ц} \cdot 0,6\sqrt{d_{\max}} = P8$, где Δr – длина цели или зоны разрыва снаряда, а $v_{ц}$ – скорость движения цели.

Программа игры "Отражение танковой атаки"

\uparrow	π	X	1	8	0	\div	P2	e^{jx}	F8	X	\uparrow
F7	XY	–	P7	F2	2	\times	e^{jx}	F4	\times	\uparrow	F7
–	P2	x^2	\uparrow	F5	–	$x < 0$	/–/	0	0	C/P	F2
C/P	BП	PO									

Перед каждым "выстрелом" в регистр X вводят угол возвышения орудия θ и нажимают клавишу C/P (первый раз B/O и C/P). После выполнения программы на индикаторе высвечивается расстояние от танка до места падения снаряда (при недолете – с отрицательным знаком) или цифры 00 при поражении танка. Условия игры можно изменять в широких пределах, изменения исходные данные. В ней могут участвовать и "танкисты", поочередно обменявшиеся выстрелами с артиллеристами.

В подобных играх цель может двигаться и под определенным углом по отношению к орудию. Представьте себя, например, командиром подводной лодки. Гидроакустическая станция Вашей подлодки обнаружила на расстоянии d корабль, движущийся со скоростью v под углом α к направлению на подводную лодку. Необходимо выпустить условную торпеду, движущуюся со скоростью v_t , под таким углом упреждения φ , чтобы поразить корабль (рис. 31). Курсы торпеды и корабля пересекаются в точке, для которой $h = b \sin \varphi = a \sin \alpha$ и $h = b_x \operatorname{tg} \varphi = (d - b_x) \operatorname{tg} \alpha$, откуда $b_x = d / (\operatorname{tg} \alpha + \operatorname{tg} \varphi) = d \sin \alpha \sin \varphi / \sin(\alpha + \varphi)$; $b = b_x / \cos \varphi = d \sin \alpha / \sin(\alpha + \varphi)$; $a = b \sin \varphi / \sin \alpha = d \sin \varphi / \sin(\alpha + \varphi)$. За время $t = b/v_t$ торпеда пройдет расстояние b , а корабль – расстояние $a' = t v = b v / v_t = m b = m d \sin \alpha / \sin(\alpha + \varphi)$. Следовательно, торпеда

пройдет по курсу корабля на расстоянии $\Delta a = a - a' = d(\sin \varphi - m \sin \alpha) / \sin (\alpha + \varphi)$, а при каждом последующем выстреле по кораблю, идущему тем же курсом, — на расстоянии $\Delta a_i = a_i - a'_i - a'_{i-1}$, где a'_{i-1} — расстояние, пройденное кораблем до предыдущего выстрела.

В соответствии с этими соотношениями составим следующую программу с исходными данными α (в радианах) = = P7, d = P8, $m=v/v_t = P4$, длиной корабля $\Delta r = P5$ и 0 = = P6.

Программа игры "Торпедная атака"

P2	\uparrow	F7	$+$	e^{jx}	F8	\div	P3	F7	e^{jx}	F4	X	
\uparrow		F3	\div	\uparrow	F6	$+$	P6	F2	e^{jx}	F3	\div	\uparrow
F6	$-$	P2	x^2	\uparrow	F5	x^2	$-$	$x < 0$	PВП	5	0	
5	C/P	F2	C/P	BП	PO							

Перед пуском очередной "торпеды" в регистр X вводят угол φ упреждения и нажимают клавишу С/П (первый раз В/О и С/П). После выполнения программы на индикаторе высвечивается расстояние от корабля до места пересечения его курса торпедой (при прохождении за корабль — с отрицательным знаком) или сигнал SOS, подаваемый кораблем при попадании в него торпеды.

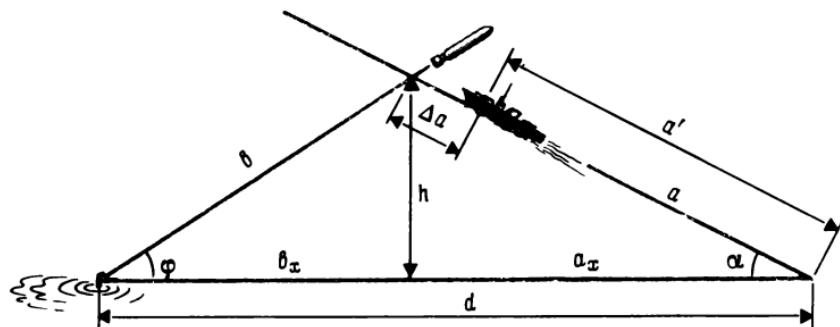


Рис. 31. Торпедная атака

Моделирование артиллерийской атаки корабля, движущегося со скоростью v_u под углом α из точки, находящейся на расстоянии d_0 (рис. 32), связано с большим числом переменных. В этом случае при выстреле из орудия с углом возвышения θ при угле упреждения φ и предельной дальности d_{\max} снаряд упадет на расстоянии $b = d_{\max} \times X \sin 2\theta$, причем за время полета снаряда $t \approx 2\sqrt{0.1d_{\max}} \times X \sin 2\theta$ корабль пройдет по курсу расстояние $a = t v_u$. Расстояние между ним и местом падения снаряда

$$r = \sqrt{a^2 + b^2 + d^2 - 2(ab \cos(\alpha - \varphi) + d(b \cos \varphi + a \cos \alpha))}.$$

Приняв для сокращения числа требуемых регистров памяти $d_{\max} = 16$ км и $2\sqrt{0.1d_{\max}} = 80$ с, составим следующую программу (v_u км/час = Р6, d км = Р7, α рад = Р8).

Программа игры "Морской бой"

P2	XY	P3	e^{jx}	X	\rightarrow	F6	X	8	0	X	\uparrow
F5	+	P5	\leftarrow	3	2	X	\uparrow	F2	\sin	X	P3
F2	\cos	X	P4	F8	e^{jx}	\rightarrow	F5	X	\uparrow	F3	-
x^2	\leftarrow	\uparrow	F5	X	\uparrow	F4	+	\uparrow	F7	-	x^2
\uparrow	\rightarrow	$+$	$\sqrt{\quad}$	C/P	BП	P0					

Перед каждым "выстрелом" выбранные значения углов θ и φ в радианах вводят соответственно в регистры Y и X и нажимают клавишу С/П (первый раз В/О и С/П).

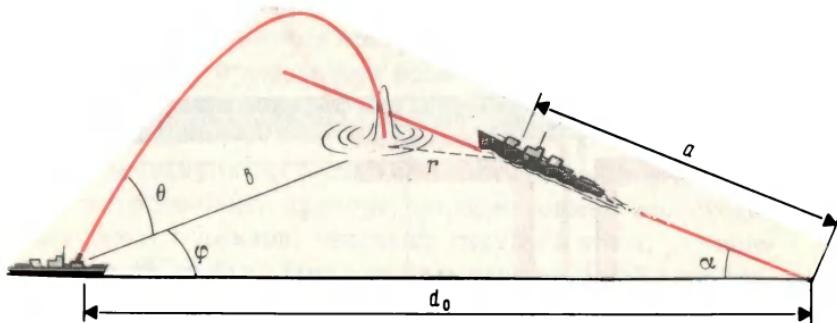


Рис. 32. Артиллерийская атака

После выполнения программы на индикаторе высвечивается расстояние от корабля до места падения снаряда, по которому и судят о качестве стрельбы. Можно вести "артиллерийскую дуэль", поочередно вводя углы θ и ϕ для своих орудий и пускай программу.

Условия подобных игр можно изменять до бесконечности, доставляя удовольствие их создателям. Примером может служить соревнование, участники которого должны пройти за кратчайшее время сложную трассу с тремя этапами: проехать на велосипеде со скоростью v_b от старта до берега реки в точке x_1 ; переплыть реку со скоростью v_p между точками x_1 и x_2 при скорости течения v_r ; бегом добраться финиша со скоростью v_b (рис. 33). Предполагается, что физическая подготовка и, следовательно, скорости движения участников одинаковы, но выигрывает участник, наиболее удачно выбравший точки x_1 и x_2 на берегах реки.

Суммарное время прохождения трассы складывается из времени езды на велосипеде $t_b = (l_1 + |x_1|)/v_b$, времени пересечения реки t_p и времени $t_b = \sqrt{(l_4 - x_2)^2 + l_3^2}/v_b$.

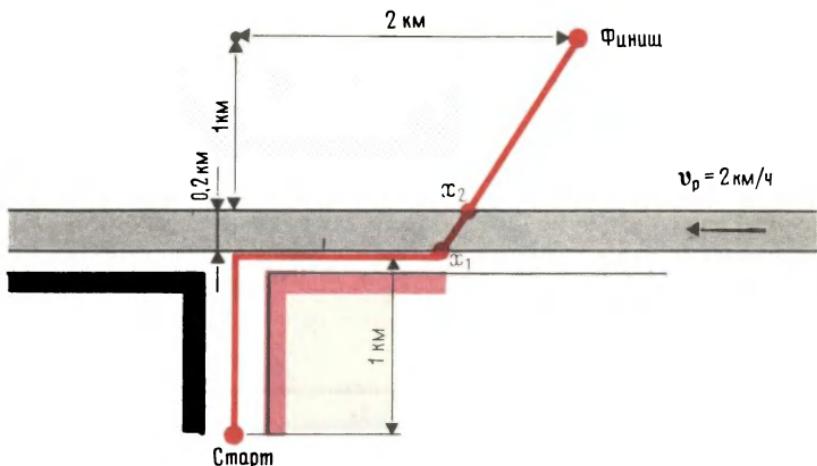


Рис. 33. Смешанный марафон

Время преодоления реки зависит от скоростей реки и пловца, но при $v_p > v_n$ пловец не сможет достичь противоположного берега в точке, расположенной правее перпендикуляра, проведенного из точки x_1 . Поэтому ограничимся случаем $v_p < v_n$, для которого $t_n = (\Delta x v_p + \sqrt{\Delta^2 x v_n^2 + (v_n^2 - v_p^2) l^2}) / (v_n^2 - v_p^2)$, где $\Delta x = x_2 - x_1$.

Будем представлять скорости в километрах в час ($v_n > 2$ км/ч), суммарное время в минутах и примем $v_p = 2$ км/ч. Тогда суммарное время

$$t = \frac{1 + \sqrt{x_1^2}}{v_b} + \frac{2\Delta x + \sqrt{(v_n \Delta x)^2 + (v_n^2 - 4)^2 / 25}}{v_n^2 - 4} + \frac{\sqrt{(2-x_2)^2 + 1}}{v_b}.$$

Роль бесстрастного судьи марафона выполняет микрокалькулятор со следующей программой ($v_b = P2$, $v_n = P3$, $v_b = P4$).

Программа игры "Смешанный марафон"

P8	XY	P7	-	2	X	→	F3	X	x^2	→	F3
x^2	4	-	P6	2	5	÷	↑	←	+	√	↑
←	+	-	↑	F6	÷	→	F7	x^2	√	1	+
F2	÷	→	F8	2	-	x^2	1	+	√	↑	F4
÷	↑	←	+	↑	←	+	6	0	X	C/P	

Каждый из участников игры вводит в регистры Y и X соответственно значения x_1 и x_2 и нажимает клавиши В/О и С/П. После выполнения программы на индикаторе высвечивается время преодоления всей дистанции. Выигрывает тот участник марафона, который показал лучшее время из трех попыток.

Микрокалькулятор с успехом может выполнять и функции навигационного прибора, определяющего положение движущихся объектов, например парусной яхты, управление которой требует большого мастерства. Чтобы приглашение участвовать в Таллинской регате не застало Вас врасплох, первые навыки в парусном спорте Вы можете приобрести, воспользовавшись следующей программой.

Программа игры "Парусная регата"

P4	F5	C/P	P5	↑	F4	—	↑	F8	X	sin	↑
√	P6	X	1	+	P7	F5	↑	F8	X	cos	↑
F6	X	1	0	×	↑	F7	÷	P7	F4	↑•	F8
X	e^{jx}	→	F7	XY	×	←	×	↑	F2	+	P2
→	↑	F3	+	P3	↑	F2	C/P	F4	C/P	БП	P0

Эта программа вычисляет текущие координаты x_i и y_i положения яхты в прямоугольной системе координат в зависимости от ее предыдущего положения (x_{i-1} и y_{i-1}), курсового угла φ и угла ориентации паруса β . Предполагается постоянный северный ветер, дующий вдоль отрицательного направления оси y (рис. 34) со скоростью v_b . В этом случае сила давления ветра на парус при скорости яхты v моделируется выражением $F = k_1(v_b \cos \beta + v \sin \beta (\varphi - \beta))^2$, где k_1 – постоянная, зависящая от площади паруса и плотности воздуха.

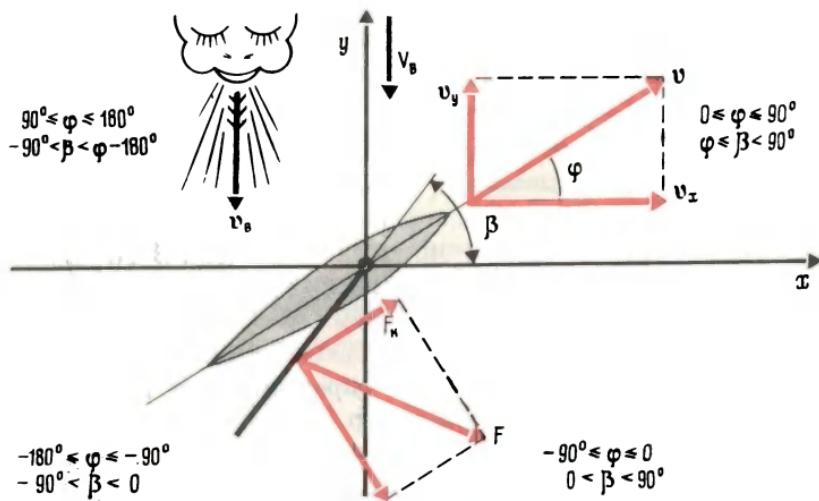


Рис. 34. Силы и скорости при движении яхты

Скорость движения яхты по курсу, зависящая от составляющей $F_k = F \sin(\beta - \varphi)$, после разгона яхты определяется формулой $v = k_2 \sqrt{F_k}$, где k_2 – постоянная, зависящая от сопротивления воды и воздуха движению яхты. Следовательно, $v = (k v_{\text{в}} \cos \beta \sqrt{\sin(\beta - \varphi)}) / (1 + k X \times \sqrt{\sin^3(\beta - \varphi)})$, где $k = k_1 k_2$. Зная v , несложно вычислить текущие координаты яхты через промежутки времени Δt (в программе принято $k = 1$, $v_{\text{в}} = 10$, $\Delta t = 1$): $x_i = x_{i-1} + v_x i \Delta t = x_{i-1} + v_i \cos \varphi_i \Delta t$; $y_i = y_{i-1} + v_y i \Delta t = y_{i-1} + v_i \sin \varphi_i \Delta t$.

Для участия в парусной регате изобразите карту акватории, где будут происходить гонки, снабдив ее координатной сеткой (рис. 35). Введите в микрокалькулятор программу, занесите в регистр 8 число $\pi/180$, в регистры 2 и 3 координаты старта x_0 и y_0 (на рис. 35 $x_0 = y_0 = 0$), введите в регистр X выбранный курсовой угол φ в градусах и нажмите клавиши В/О и С/П. После выполнения программы высветится угол β в градусах, который Вы можете сохранить или изменить. Последующее нажатие клавиши С/П приведет к вычислению координат x_1 и y_1 яхты, хранящихся в регистрах X и Y (а также 2 и 3) соответственно. Отметив на карте эти координаты, нажмите клавишу С/П, что приведет к высвечиванию установленного ранее угла φ . Изменив его, нажмите клавишу С/П, что приведет к высвечиванию угла β . Сохранив или изменив его, снова нажмите клавишу С/П, что приведет к высвечиванию координаты x_2 и вычислению y_2 , хранящейся в регистре Y . Продолжая подобным образом, проведите яхту до контрольного пункта. Например, в гонке по трассе, показанной на рис. 35, яхтсмен, выйдя из пункта A , должен отшвартоваться у причала B и вернуться в точку старта, затратив минимальное число "шагов" и не посадив яхту на прибрежные рифы.

Приведем пример прохождения начального участка трассы, показанной на рис. 35 (указанные в скобках ко-

ординаты для упрощения их регистрации округлены до десятых долей): C_x P2 P3 20 В/О С/П 50 С/П (3,2 XY 1,1) С/П (20) С/П (50) С/П (6,3 XY 2,3) С/П (20) С/П (50) С/П (9,5 XY 3,4) С/П (20) 50 С/П (50) 60 С/П (10,7 XY 4,9) С/П (50) С/П (60) 65 С/П (11,9 XY 6,4) С/П (50) 120 С/П (65) 70 /-/ С/П (11,3 XY 7,5) . . . На прохождение всей трассы, включая и обратный путь, было затрачено 26 шагов. Попытайтесь пройти эту трассу быстрее.

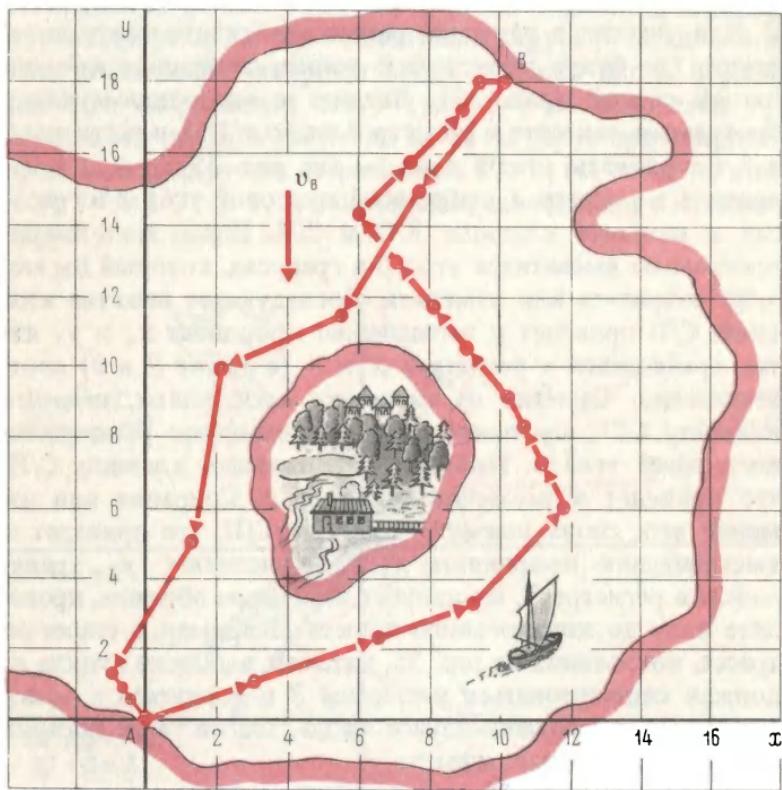


Рис. 35. Карта акватории для парусных гонок

Следует заметить, что яхта будет двигаться в выбранном направлении только при правильной ориентации паруса под углом β , значения которого для движения в каждом из квадрантов приведены на рис. 34.

Микрокалькулятор с успехом может выполнять и функции реле времени, что представляет интерес для большой армии фотолюбителей. Программа использования микрокалькулятора "Электроника Б3-21" в качестве реле времени выглядит следующим образом:

НОП Р2 \uparrow F3 – $x < 0$ Р \uparrow F2 С/П В/О

Заносим в регистр 3 специально подобранное число (например, 1,0001), после чего, набрав в регистре X нужную выдержку в секундах (например, 20), нажимаем клавишу С/П и ждем окончания работы программы. Чтобы программа окончилась, требуется 20 циклов, так как $20 - 1,0001 \times 20 < 0$. Время выполнения программы оказалось равным 13 с и, значит, один цикл выполняется за $13/20 \approx 0,65$ с. Чтобы это время равнялось 20 с, требуется $20/0,65 \approx 31$ цикл. А чтобы при вводе числа 20 выполнился 31 цикл, следует занести в регистр 3 число $20/31 \approx 0,64$. Оператор НОП нужен, чтобы время выполнения программы было одинаковым при первом и последующих пусках программы. Поэтому после оператора С/П был записан оператор В/О, передающий управление по адресу 02.

Важно подчеркнуть для фотолюбителей, что погрешность малых интервалов времени накапливается, так как время выполнения программы $t = kN + t_0$, где N – набираемое на индикаторе число, а k – постоянная, равная отношению времени выполнения цикла $t_0 \approx 0,65$ с к содержимому регистра 3. Поэтому если подобрать $k = 1$ для больших интервалов времени, то относительная погрешность $\Delta t/t \approx t_0/t$ и, например, для $t = 1$ с она составит примерно 65 %.

3.5. Берегите автомобиль!

Составляя программы предыдущего раздела, мы пользовались готовыми расчетными формулами, представляющими собой упрощенные математические модели, не учитывающие многие реальные факторы, например сопротивление воздуха при полете снаряда. Желая получить более точные результаты, необходимо при решении задач учитывать реальные факторы. Так как сила сопротивления воздуха при полете снаряда приближенно пропорциональна квадрату скорости и направлена против вектора скорости v , это позволяет согласно третьему закону Ньютона уточнить уравнение движения снаряда, записав его в векторной форме $ma = F - kv|v|$, где k – постоянная, определяемая формой снаряда и плотностью воздуха, а $F = mg$ – направленная к центру Земли сила тяжести, выражаемая через массу m снаряда и ускорение $g \approx 10 \text{ м/с}^2$ свободного падения (рис. 36).

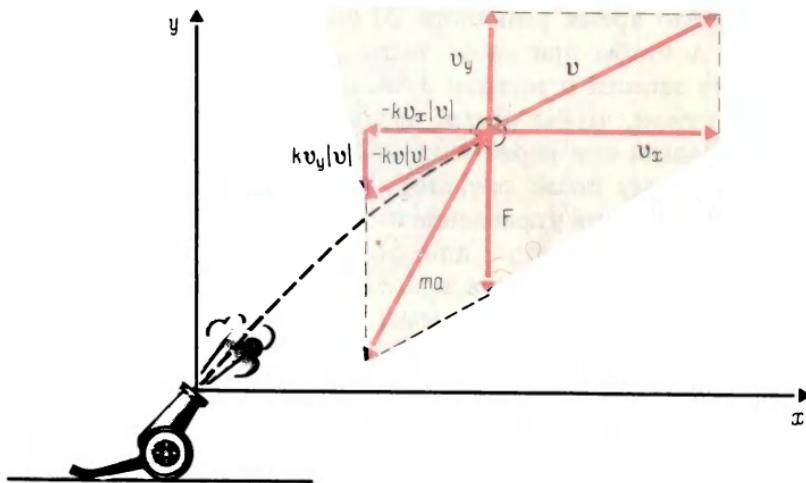


Рис. 36. Силы и скорости при полете снаряда

Учитывая, что ускорение $a = dv/dt$, составим уравнение $dv/dt = g - k v|v|/m$, содержащее неизвестную величину v и ее производную (такие уравнения называют дифференциальными). Чтобы избавиться от векторов, введем прямоугольную систему координат, ось абсцисс x которой совмещаем с плоскостью Земли, а ось ординат y направим вертикально вверх. Тогда для проекций v_x и v_y скорости v исходное векторное уравнение в системе из двух скалярных:

$$\frac{dv_x}{dt} = -\frac{k v_x}{m} \sqrt{v_x^2 + v_y^2}, \quad \frac{dv_y}{dt} = -g - \frac{k v_y}{m} \sqrt{v_x^2 + v_y^2}.$$

Решить эту систему уравнений точно не удается, так как нелинейные дифференциальные уравнения (содержащие нелинейные функции искомой переменной или ее производной) обычно точно не решаются — другими словами, не существует формул с конечным числом членов, являющихся их решением.

В подобных случаях прибегают к приближенным методам решения, среди которых при вычислениях на ЭВМ чаще всего предпочитают методы цифрового моделирования. В этих методах время протекания процесса разбивают на достаточно большое число одинаковых интервалов Δt , позволяющее считать процесс неизменным в пределах такого интервала. Решая упрощенные уравнения для каждого из интервалов и учитывая изменение условий при переходе к следующему интервалу, получают "табличную" модель протекания процесса $y_i = y(x_i)$, где $i = 0, 1, 2, \dots$ — номера интервалов. Программные реализации таких моделей на входных языках ЭВМ называют цифровыми моделями.

Рассмотрим применение цифрового моделирования для расчета траектории полета снаряда с учетом сопротивления воздуха. Прежде всего заменим в уравнениях производные приближенными соотношениями $dv/dt = ((v(t + \Delta t) -$

$-v(t)/\Delta t$, методическая погрешность которых уменьшается при уменьшении Δt . Пронумеровав значения v от v_0 в момент $t = t_0$ до v_i при $t_i = t_0 + i\Delta t$, запишем $dv/dt = (v_{i+1} - v_i)/\Delta t$. Это позволяет заменить исходную систему уравнений приближенными (называемыми разностными), в правой части которых с учетом $v_{i+1} \approx v_i$ значения v_{i+1} заменены на v_i или $v_{xi+1} = v_{xi} - k\Delta t v_{xi} \times \sqrt{v_{xi}^2 + v_{yi}^2}/m$; $v_{yi+1} = v_{yi} - g\Delta t - k\Delta t v_{yi} \sqrt{v_{xy}^2 + v_{yi}^2}/m$. Погрешность решения этих уравнений зависит от выбора Δt , методическая погрешность снижается при уменьшении Δt , но увеличивается число $n = t/\Delta t$ шагов решения и, следовательно, операционная ошибка. Поэтому существует оптимальное значение Δt , при котором полная погрешность решения разностных уравнений минимальна.

При выбранном значении Δt сократить время вычислений и повысить точность решения удается при снижении методической погрешности. Координаты снаряда в момент времени $t_i = t_0 + i\Delta t$ связаны со скоростью v соотношениями $x_{i+1} = x_i + v_{xi}\Delta t$; $y_{i+1} = y_i + v_{yi}\Delta t$ или, с учетом $v_i \approx v_{i+1}$, формулами $x_{i+1} = x_i + v_{xi+1}\Delta t$; $y_{i+1} = y_i + v_{yi+1}\Delta t$. Если усреднить скорость на интервале $v_{cpi} = (v_{i+1} + v_i)/2$, то для расчета координат проще воспользоваться формулами $x_{i+1} = x_i + (v_{xi} + v_{xi})\Delta t/2$; $y_{i+1} = y_i + (v_{yi+1} + v_{yi})\Delta t/2$.

Приведенные соображения позволяют приступить к составлению цифровой модели полета снаряда, но вызывает сомнение выбор постоянной k . Так как она зависит от плотности воздуха, уменьшающейся с высотой, то ее можно учесть формулой $k_i \approx k_0/(1 + \alpha)$, где α – некоторая постоянная. Наиболее удобными единицами измерения расстояния и скорости представляются метры и метры в секунду, а для простоты вычислений удобно принять $\Delta t = 1$ с.

Распределив память микрокалькулятора "Электроника Б3-21" для хранения исходных и текущих данных ($x_i =$

$= P2$, $y_i = P3$, $v_{xi} = P4$, $v_{yi} = P5$, $k_0/m = P6$, $\alpha = P7$), составим следующую программу.

Цифровая модель полета снаряда

F3	\uparrow	F7	X	1	+	\leftarrow	F5	x^2	\uparrow	F4	x^2
+	$\sqrt{ }$	\uparrow	F6	X	\uparrow	\rightarrow	\div	\uparrow	F5	X	-
F4	XY	X	-	P4	+	2	\div	\uparrow	F2	+	P2
\rightarrow	1	0	+	\uparrow	F5	XY	-	P5	+	2	\div
\uparrow	F3	+	P3	\uparrow	F2	C/P	БП	РО			

После ввода этой программы следует в регистры 2 и 3 соответственно занести координаты x_0 и y_0 точки вылета снаряда из орудия, в регистры 4 и 5 – составляющие $v_{x_0} = v \cos \theta$ и $v_{y_0} = v_0 \sin \theta$, а в регистры 6 и 7 – выбранные значения постоянных k_0/m и α . После нажатия клавиш В/О и С/П программа вычисляет (время счета около 10 с) и заносит в регистры X и Y соответственно значения x_1 и y_1 . Затем достаточно нажимать клавишу С/П и регистрировать очередные значения x_i и y_i , по которым легко построить баллистическую кривую полета снаряда при различных исходных условиях (θ , v_0 , k_0/m и α).

Выполняя расчет по составленной программе, обнаруживаем, что возможно изменение параметров моделируемого процесса после каждого или нескольких отсчетов. Таким образом, можно моделировать полет не только снаряда, но и различных управляемых объектов. Это навело нас на мысль создания домашней школы автолюбителей.

При движении автомобиля по прямой в горизонтальной плоскости на него действует сила тяги двигателя F (или усилие торможения $F < 0$) и сила сопротивления воздуха $F_c \approx -k v |v|$, направленная против движения и примерно пропорциональная квадрату скорости. Следовательно, уравнение движения принимает вид $ma = F - k v |v|$, где m – масса автомобиля. Если не учитывать задний ход и

ограничиться случаем $v \geq 0$, то сила сопротивления воздуха всегда будет неположительной и уравнение движения упрощается: $ma = F - kv^2$.

Читатель, вероятно, думает, что забыта сила трения? Нет, просто оказалось, что при движении со скоростью более 20 км/ч по ровному шоссе с хорошим покрытием эта сила намного меньше силы сопротивления воздуха и ею можно пренебречь (при желании силу трения легко учесть уменьшением тяги или увеличением силы торможения).

Из составленного уравнения движения легко найти и установившуюся скорость движения при неизменной силе тяги $v_{\text{уст}} = \sqrt{F/k}$. Заменив уравнение движения разностным $v_{i+1} = v_i + (F - kv_i^2)\Delta t/m$, мы попытались уменьшить методическую погрешность, возникающую при подстановке вместо изменяющейся скорости ее значения в начале каждого i -го интервала Δt .

Прикинув, ошибку по знаку при вычислении приращения скорости получим $\Delta v_{i+1} = v_{i+1} - v_i$. При разгоне автомобиля за время Δt его скорость увеличится и вычисления по формуле $\tilde{\Delta}v_{i+1} = (F - kv_i^2)\Delta t/m$ приведут к завышенному результату. Для уменьшения ошибки воспользуемся первым приближением скорости $v_{i+1} = v_i + \tilde{\Delta}v_{i+1}$ и повторим вычисление $\tilde{\Delta}v_{i+1} = (F - k\tilde{v}_{i+1}^2)\Delta t/m$. Но теперь в формуле фигурирует значение \tilde{v}_{i+1} , которое, как мы установили, превышает истинное. Это снова приведет к ошибке, но уже с другим знаком — приращение скорости $\tilde{\tilde{\Delta}}v_{i+1}$ окажется меньше истинного. А так как "истина лежит посередине", то разумно принять $v_{i+1} = v_i + (\tilde{\Delta}v_{i+1} + \tilde{\tilde{\Delta}}v_{i+1})/2 = v_i + (2F + k(v_i^2 + \tilde{v}_{i+1}^2)) \cdot \Delta t / 2m$, где $\tilde{v}_{i+1} = v_i + (F - kv_i^2)\Delta t/m$.

Читатель может самостоятельно убедиться, что и при торможении также произойдет частичная компенсация методической погрешности определения скорости.

Описанный простейший метод уменьшения методической погрешности решения дифференциальных уравнений носит название исправленного метода Эйлера. Кроме него известно множество других, сведения о которых и сравнительную их оценку при вычислениях на микрокалькуляторах читатель может найти в [7].

Вычислив, как и в предыдущей задаче, приращение пройденного пути по формуле $\Delta s = (v_i + v_{i+1})\Delta t/2$, несложно определить координаты автомобиля $x_{i+1} = x_i + \Delta s \cos \psi$; $y_{i+1} = y_i + \Delta s \sin \psi$ при его движении под углом ψ (рис. 37, а). При повороте передних колес на угол φ автомобиль начнет двигаться по окружности с радиусом $r \approx h/\tan \varphi$, где h — расстояние между осями передних и задних колес (рис. 37, б). При $\varphi < 1$ для простоты можно принять $\tan \varphi = \varphi$ и $r \approx h/\varphi$. Если при повороте автомобиль движется со скоростью v , то за время Δt он опишет дугу длиной $\Delta s \approx v\Delta t$, а направление движения изменится на угол $\Delta\psi = \Delta s/r = v\Delta t \varphi/h = \gamma v \varphi$, где $\gamma = \Delta t/h$ — постоянный множитель.

Допустим, что в момент времени t_i автомобиль из точки с координатами x_i и y_i движется в направлении ψ_i , а угол поворота равен φ . Где он окажется в момент времени $t_{i+1} = t_i + \Delta t$ и в каком направлении будет двигаться?

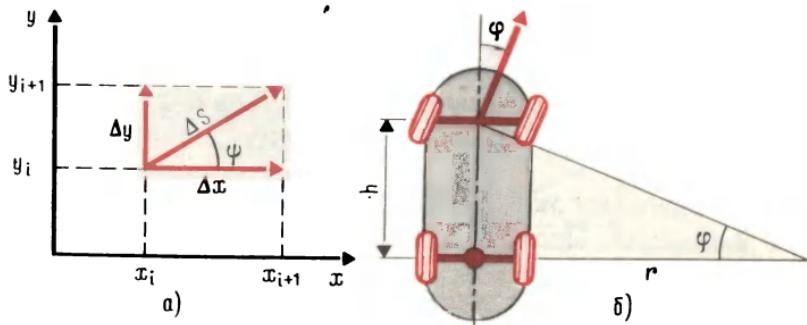


Рис. 37. Моделирование движения автомобиля

Ответ на первый вопрос практически найден — новое направление $\psi_{i+1} = \psi_i + \Delta\psi = \psi_i + \gamma v \varphi$. Для ответа на второй вопрос рассмотрим траекторию движения автомобиля (рис. 38, а), предположив, что за время Δt его скорость, изменившись по направлению на угол $\Delta\psi$, не изменяется по величине. Для простоты заменим движение по дуге движением по ломаной прямой из точки x_i, y_i в точку x', y' , а затем — в точку x_{i+1}, y_{i+1} (рис. 38, а). Такое упрощение приведет к небольшой погрешности, если длина дуги мала ($\Delta s \ll r$), но тогда $\Delta x = \Delta x_1 + \Delta x_2 = \Delta s_1 \cos \psi_i + \Delta s_2 \cos \psi_{i+1}$; $\Delta y = \Delta y_1 + \Delta y_2 = \Delta s_1 \sin \psi_i + \Delta s_2 \sin \psi_{i+1}$ и при $\Delta s_1 = \Delta s_2 = v\Delta t/2$ получим $\Delta x = v\Delta t (\cos \psi_i + \cos \psi_{i+1})/2 = v\Delta t \cos(\Delta\psi/2) \cos(\psi_i + \Delta\psi/2)$; $\Delta y = v\Delta t (\sin \psi_i + \sin \psi_{i+1}) = v\Delta t \cos(\Delta\psi/2) \times \sin(\psi_i + \Delta\psi/2)$. Если выполняется соотношение $\Delta\psi \ll 1$, то можно принять $\cos \Delta\psi/2 \approx 1$ и формулы для вычисления координат примут вид $x_{i+1} = x_i + v\Delta t \cos(\psi_i + \Delta\psi/2)$; $y_{i+1} = y_i + v\Delta t \sin(\psi_i + \Delta\psi/2)$.

При резком повороте на большой скорости автомобиль может "занести" или даже опрокинуть. Для моделирования опрокидывания достаточно на каждом шаге сравнивать центробежную силу $F_u = mv^2/r$ и опрокидыва-

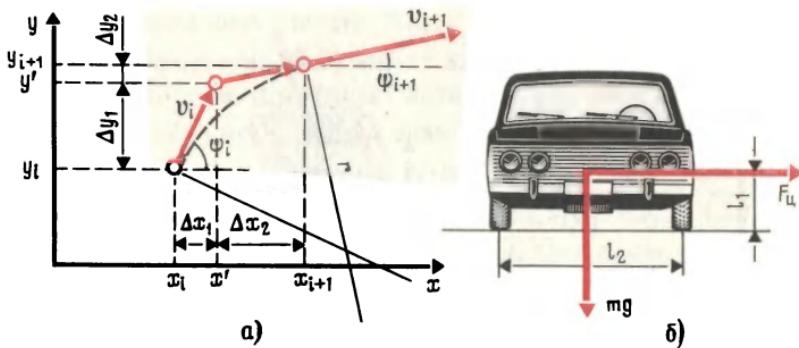


Рис. 38. Моделирование поворотов автомобиля

ющий момент $M_{\text{опр}} = F_{\text{ц}} l_1$, где l_1 — высота центра тяжести автомобиля (рис. 38, б). Чтобы автомобиль с шириной l_2 колеи не перевернулся, должно выполняться соотношение $|M_{\text{опр}}| < mgl_2/2$ или $|\varphi v^2| < B$, где $B = h g l_2/2$ — постоянный коэффициент.

"Занос" автомобиля моделировать значительно сложнее и, учитывая, что в современных автомобилях имеются специальные устройства для уменьшения заноса, ограничимся моделированием опрокидывания, составив следующую программу.

Цифровая модель движения автомобиля

P2	F8	C/P	P8	F7	1	+	P7	F6	9	÷	X
—	↑	F2	+	↑	F6	+	2	÷	P6	x^2	x^2
$1/x$	↑	F8	x^2	—	$\sqrt{\quad}$	F6	↑	→	F8	X	↑
F3	+	P3	e^{jx}	←	XY	×	→	×	↑	F4	+
P4	F5	↑	←	+	P5	↑	F4	C/P	БП	P0	

По этой программе вначале вычисляется при заданном значении сил тяги (или торможении) F , хранящемся в регистре 2, и постоянном коэффициенте $k \Delta t/m = 1/9$ скорость автомобиля по формулам¹ $v_{i+1} = (1 - v_i/9)v_i + F \Delta t/m$; $v_{i+1} = (v_i + v_{i+1})/2$ и при $B = 1$ оценивается устойчивость автомобиля по условию $(1/v^2)^2 - \varphi^2 > 0$. Если это условие не выполняется, то возникает переполнение, свидетельствующее об "опрокидывании" автомобиля.

Затем по программе вычисляется новое направление $\psi_{i+1} = \psi_i + \psi v$ по $\psi_i = P3$ и заданному "водителем" углу $\varphi = P8$, а также координаты $x_{i+1} = x_i + (v_i + \tilde{v}_{i+1}) \cos X \times (\psi_i + v\varphi)/2$; $y_{i+1} = y_i + (v_i + \tilde{v}_{i+1}) + \sin(\psi_i + v\varphi)/2$ по значениям $x_i = P4$, $y_i = P5$. Регистр 6, хранящий текущее значение скорости, выполняет функции спидометра, а

¹ Для большего "игрового эффекта" эти формулы отличаются от классических и "исправлений" формул Эйлера для "затягивания" разгона и торможения автомобиля.

регистр 7 — "часов", так как его содержимое равно числу n шагов, а $t_n = n\Delta t$.

При участии в "авторалли" на листе бумаги с координатной сеткой следует изобразить трассу автопробега (пример показан на рис. 39), ввести в микрокалькулятор программу, очистить регистры 6 и 7, занести в регистры 4 и 5 координаты старта x_0 и y_0 , а в регистр 3 — начальное направление движения радиан ψ_0 . Теперь можно "нажимать на газ", занося в регистр X выбранную силу тяги $-1 \leq F \leq 1$

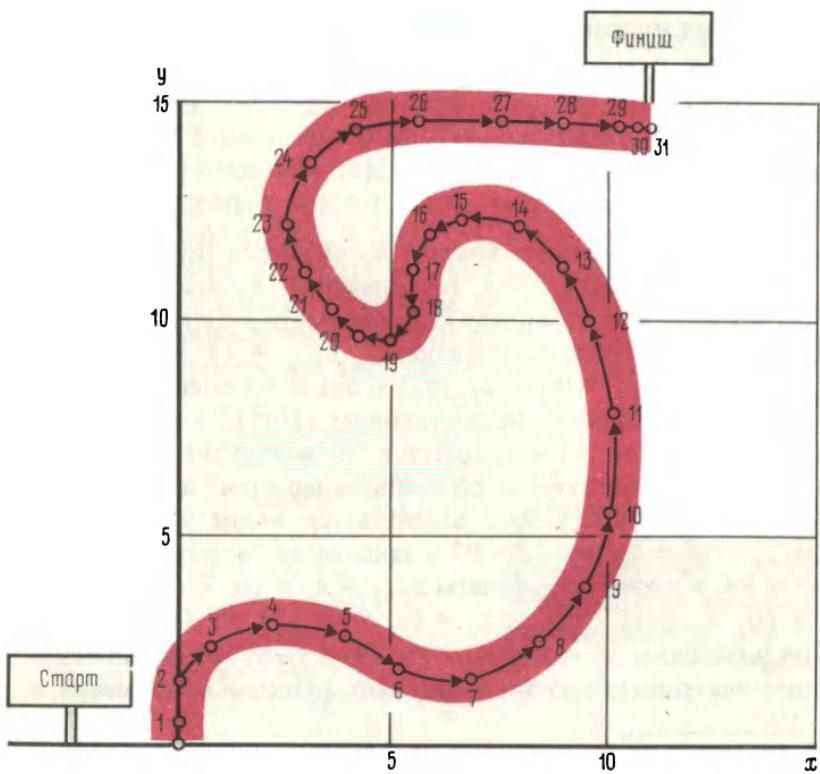
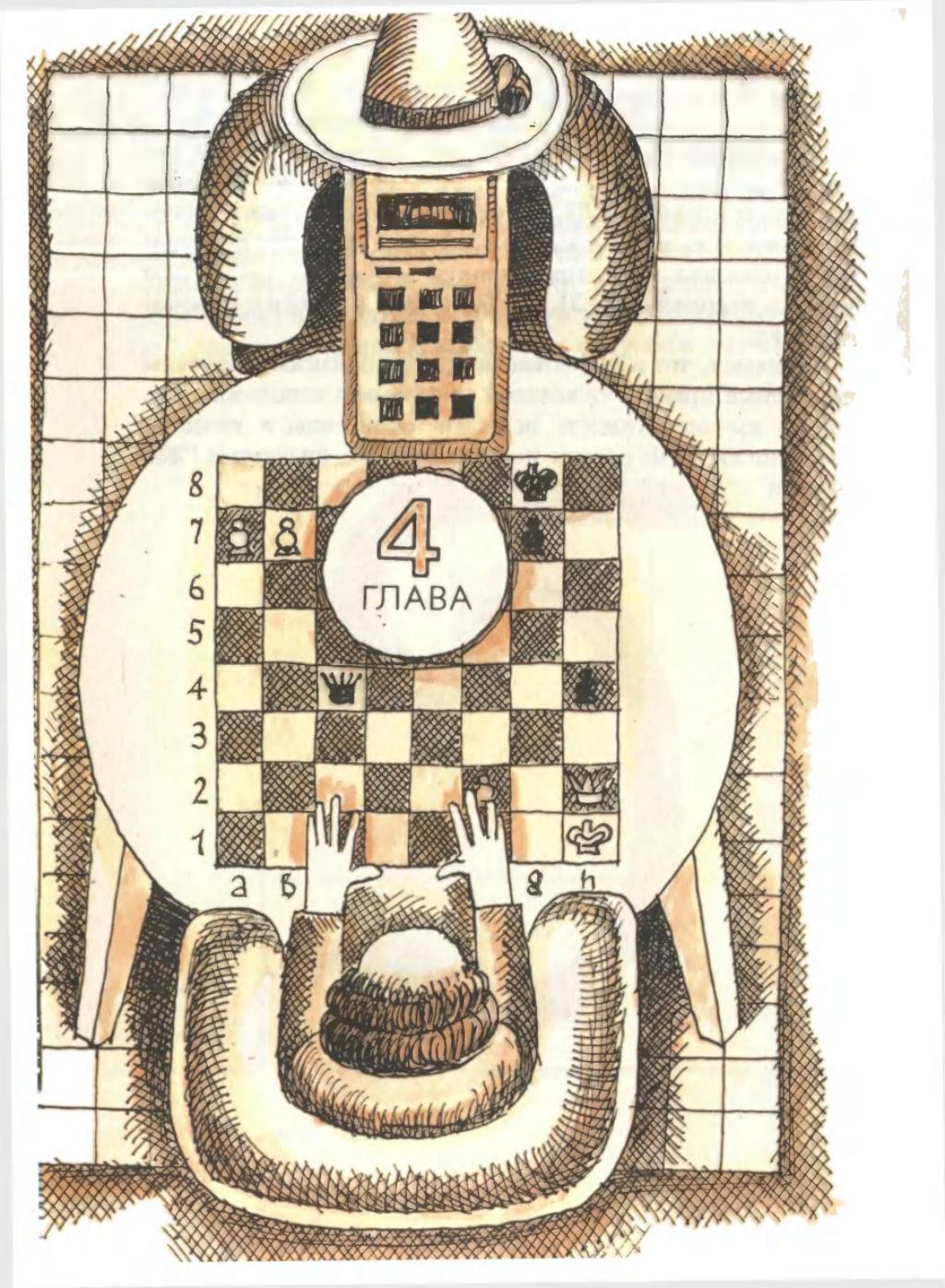


Рис. 39. Карта автопробега

(или впоследствии, силу торможения), и нажать клавиши В/О и С/П. После выполнения программы на индикаторе высветится угол φ положения рулевой колонки, после изменения которого $-1 \leq \varphi \leq 1$ или сохранения прежним нажимают клавишу С/П, что приведет к вычислению координат $FX = x_1$ и $FY = y_1$. Аналогичные операции повторяют на каждом шаге. Приведенную на рис. 39 трассу нам удалось преодолеть за 31 шаг без съезда с дороги и опрокидывания.

Надеемся, что и Вы, объяснив своим близким и друзьям несложные правила описанной программы использования, дадите им возможность испытать свои силы в качестве автогонщиков, не рискуя ни их здоровьем, ни своими "Жигулями".



4

ГЛАВА

МИКРОКАЛЬКУЛЯТОР, ВАШ ХОД!

МОЖЕТ ЛИ МЫСЛИТЬ МИКРОКАЛЬКУЛЯТОР?

ИГРЫ С КАМЕШКАМИ

МИКРОКАЛЬКУЛЯТОР ИГРАЕТ В ШАХМАТЫ

"КОКТЕЙЛЬ" ИЗ СТРАТЕГИЙ

ИГРЫ СО СЛУЧАЕМ

4.1. Может ли мыслить микрокалькулятор?

Этот вопрос не риторический, так как лишь несколько десятилетий назад люди горячо спорили о том, могут ли мыслить машины. Причиной дискуссии явились первые успехи ЭВМ в решении задач, относимых ранее к чисто интеллектуальной деятельности человека — переводах текстов с иностранных языков, сочинениях стихов и музыки, решениях шахматных этюдов, диагностике болезней. Казалось, что достаточно построить ЭВМ с емкостью памяти, не уступающей человеческой, и искусственный мозг решит все проблемы.

С тех пор сменилось несколько поколений ЭВМ. Современные супермашины выполняют вычисления со скоростью, о которой не мечтали даже фантасты, а в памяти таких машин хранится колossalный объем информации. Тем не менее ЭВМ пока не стала чемпионом мира по шахматам, а специалисты по искусственноому интеллекту вынуждены признать, что до его создания все еще далеко.

Так все таки, может ли мыслить ЭВМ? Сегодня на эту тему не спорят — оказалось, что нельзя дать строгого определения понятию "мыслить" и, следовательно, спорить не о чем. Однако это не означает, что работы по развитию искусственного интеллекта не имеют смысла. Наоборот, накопление знаний о способах решения на ЭВМ задач, характерных для умственной деятельности человека, помогает понять, как работает человеческий мозг и каким образом решать на машине задачи, которые способны решать люди.

Умственная деятельность человека со дня его рождения связана с постановкой и решением задач моделирования причинно-следственных связей в окружающем мире, а накопление в памяти способов решения таких задач и называет-

ся жизненным опытом. Встречаясь с очередной задачей, человек использует готовое решение или, при его отсутствии, ищет решение по общим чертам (ассоциациям) с хранящимися в памяти алгоритмами решения других задач, абстрагируясь от их конкретных приложений. Следовательно, можно условно выделить два взаимосвязанных вида умственной деятельности человека — рефлексивную, для которой характерно использование закрепленных в памяти навыков ("программ") решения задач, и ассоциативную, связанную с построением алгоритмов решения новых задач в процессе творческого осмысливания и преобразования хранящейся в памяти информации.

Решение проблемы искусственного интеллекта в свое время началось с имитации рефлексивной деятельности мозга, простейшим аналогом которой является "обучение" ЭВМ решению задач вводом в ее память программ автоматических вычислений. Кстати, именно поэтому в зарубежных программируемых микрокалькуляторах оператор перехода в режим программирования обычно обозначают сокращением *LRN* от слова *learn* — "учить".

Более сложными аналогами рефлексивного мышления можно считать самообучающиеся машинные программы. Еще на заре кибернетики приобрела известность програм-

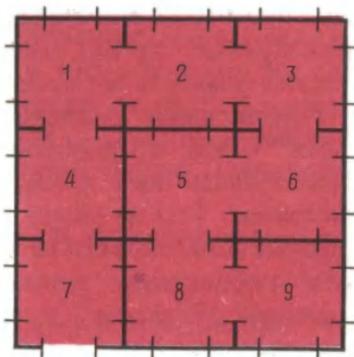


Рис. 40. Схема "лабиринта"

ма отыскания методом проб и запоминания выхода из лабиринта. Схема одного из таких "лабиринтов" изображена на рис. 40, где помещение с номером 9 имеет одну внутреннюю дверь, а остальные помещения — две двери, причем одно из них имеет одну наружную дверь. Программа должна найти и запомнить последовательность 74123658 номеров комнат на искомом пути.

Совсем недавно о подобных программах можно было прочитать лишь в книгах, но сегодня с рассмотренной задачей успешно справляется микрокалькулятор "Электроника Б3-21", запомнивший следующую программу.

Самообучающаяся программа

P2	F3	x=0	F,	F2	x=0	3	F4	1	+	P4	1
0	-	x=0	P3	1	P4	F4	БП	,	F4	9	-
x=0	P/-/	1	P3	F5	C/П	БП	P0	F5	1	0	X
↑	F4	+	P5	БП	F1						

После ввода этой программы следует очистить регистры 3, 4 и 5. Микрокалькулятор после нажатия клавиши С/П (первый раз В/О и С/П) генерирует очередной пробный номер от 1 до 9, но не имея "органов чувств", нуждается в помощи пользователя — последний должен очистить регистр X нажатием клавиши C_x, если высвечиваемый номер не соответствует номеру очередного помещения на искомом пути. Если очередным оказалось помещение 9, то при последующих пусках микрокалькулятор высвечивает последовательность номеров комнат на пути к комнате 9.

Рефлексивная деятельность мозга протекает "механически" — идя по знакомой дороге, мы не задумываемся, куда ставить ноги и обычно обдумываем совсем другие проблемы. Тем не менее при этом мы не сталкиваемся с другими пешеходами и не попадаем под автомобили — наш мозг рефлексивно решает задачу движения. Поэтому с понятием "мыслить" в основном связана ассоциативная деятельность мозга, а задача создания искусственного ин-

теллекта заключается в разработке методов имитации на ЭВМ именно этой деятельности мозга.

Наиболее общий метод построения (синтеза) оптимальной структуры алгоритма решения задачи заключается в полном переборе всех путей решения и выборе среди них оптимального. Однако трудоемкость этого метода быстро возрастает при увеличении числа возможных вариантов структуры, например число всех вариантов шахматной игры по оценкам математиков составляет около $2 \cdot 10^{116}$ *. Поэтому невозможно построить ЭВМ, играющую в шахматы методом полного перебора — для этого просто не хватит материала.

Тем не менее сегодня в шахматы играют не только люди, но и машины, хотя и с меньшим успехом. Это оказалось возможным с развитием эвристических методов составления алгоритмов шахматной игры. Слово "эвристика" переводят с греческого (вспомните знаменитое восклицание Архимеда "Эврика!") как "искусство нахождения истины" — совокупность догадок, логических приемов и методических правил поиска путей решения задачи без полного перебора всех возможных вариантов.

Эвристический метод подзадач (решение с конца) заключается в исследовании известного результата решения с целью выделения необходимых для его достижения промежуточных результатов и разбиения исходной задачи на отдельные этапы (подзадачи). Достаточно общим является и метод аналогий (близкий к индуктивному методу "от частного к общему"), основанный на такой формулировке условий задачи, при которой становится очевидной возможность ее решения по примеру других (аналогичных) задач.

Рассмотрим следующую задачу: на одной из трех тар-

* Н. Кобринский, В. Пекелис. Быстрее мысли. — М.: Молодая гвардия, 1959.

лок сложена горка из n блинов различного размера, которую требуется переложить по одному блину на другую тарелку за минимальное число перемещений, причем на любой из тарелок большие блины запрещается класть на меньшие. Ограничимся вначале случаем $n = 3$, обозначив конечные и промежуточные состояния решения задачи последовательностями ABC , в которых символы блинов A , B и C заменены номерами тарелок, на которых находятся соответствующие блины. Тогда исходное состояние (рис. 41, а) будет закодировано символом 111, а конечное (рис. 41, б) – символом 333.

При полном переборе от исходного состояния 111 перемещением верхнего блина A можно перейти только к состояниям 211 или 311. Продолжая подобный перебор возможных переходов в очередные состояния, удобно отмечать их вершинами графа, ветви которого соответствуют переходам. Совмещая в таком графе одинаковые вершины, представим его в компактной форме (рис. 41, в) и найдем кратчайший и, следовательно, оптималь-

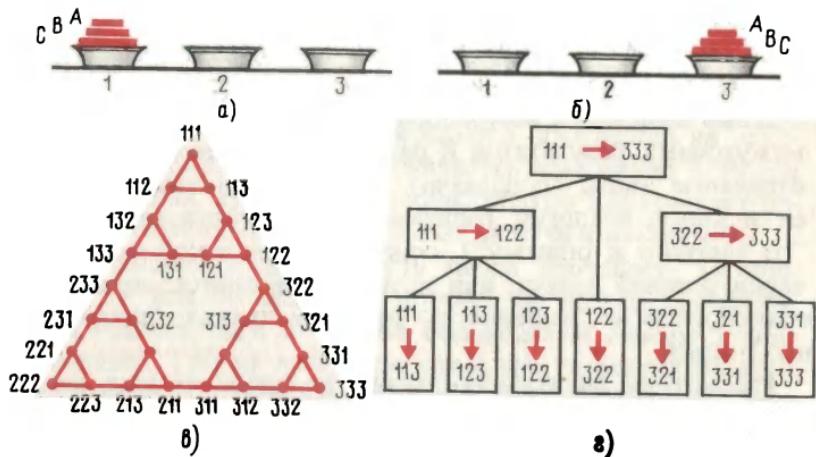


Рис. 41. К задаче о блинах

ный по условию задачи путь решения, отмеченный на графе жирной линией.

Попытаемся упростить поиск оптимального решения, используя метод подзадач. Анализируя возможные переходы к заданному конечному состоянию (рис. 41, б), легко заметить, что перед перемещением наибольшего блина из тарелки 1 на тарелку 3 блины A и B могут лежать только на тарелке 2 (состояние 221). При этом по условию задачи верхним должен быть блин A и, следовательно, предыдущим должно быть состояние 321. Таким образом, задача разбивается на подзадачи переходов $111 \rightarrow 321$, $321 \rightarrow 221$, $221 \rightarrow 333$. Разбивая первую и третью подзадачу на простейшие (перемещения одного блина), находим оптимальное решение, этапы поиска которого представлены графом на рис. 41, г.

Решение задачи о блинах рассмотренным способом усложняется при $n > 3$. Между тем установлено [1], что минимальную последовательность $2^n - 1$ перемещений объектов, требуемую для решения рассматриваемой задачи, можно определить удивительно просто. Для этого следует представить порядковые номера N очередных перемещений в двоичном коде $\dots a_4 a_3 a_2 a_1 a_0$, где символы 0 или 1 в двоичном разряде a_i соответствуют $N = \dots \dots a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$. Если обозначить двоичные разряды a_i , начиная с младшего, символами блинов A , B и C , то первый справа разряд, содержащий единицу, будет определять перемещаемый блин, причем четные (отсчитывая сверху в исходном положении) перемещаются на тарелки с последовательностью номеров 2, 3, ..., а нечетные — с последовательностью 3, 2, Следовательно, алгоритм оптимального решения задачи о блинах при $n = 3$ отображается табл. 10.

Составив для микрокалькулятора программу преобразования номера N в двоичный код, получим программу оптимального решения задачи о блинах. Для микрокаль-

кулятора "Электроника Б3-21" можно использовать следующую программу, представляющую вводимые в регистр X числа от 0 до 255 в двоичном коде и позволяют решать рассматриваемую задачу при числе блинов $n \leq 8$.

Программа представления чисел в двоичном коде

P2	1	2	8	P3	C_x	P8	F8	1	0	X	P8
F2	↑	F3	—	$x \geq 0$	÷	P2	F8	1	+	P8	F3
2	÷	P3	1	—	$x < 0$	F1	F8	C/P	BП	P0	

Эта программа пригодна для решения не только задачи о блинах, но и множества других задач, этапы решения которой представимы последовательностью двоичных кодов порядковых чисел. Если бы удалось создать подобную программу для всех задач, то была бы решена, возможно, и проблема искусственного интеллекта.

Многие эвристические методы связаны с творческим представлением человеком "в воображении" возможных изменений структуры алгоритма решения задачи. Образное мышление у большинства людей относится в основном к функциям правого полушария мозга, тогда как левое ведает логическим мышлением, связанным с формирови-

Таблица 10

Алгоритм решения задачи о блинах

N	C	B	A	Перемещаемый блин	Положение блинов
0	0	0	0	—	111
1	0	0	1	A	311
2	0	1	0	B	321
3	0	1	1	A	221
4	1	0	0	C	223
5	1	0	1	A	123
6	1	1	0	B	133
7	1	1	1	A	333

рованием последовательностей имеющих смысл символов [5]. А так как в современных ЭВМ обрабатываемая информация также представляется последовательностями символов, то с помощью таких машин удается в некоторых пределах имитировать деятельность лишь левой половины мозга, что существенно ограничивает "мыслительные" способности ЭВМ.

В задаче о блинах мы располагали полной информацией, позволившей построить все возможные пути решения и выбрать среди них оптимальный. Однако в условиях большинства практических задач полная информация отсутствует и приходится искать оптимальные решения на каждом этапе в условиях неопределенности последствий таких решений. Наше исследование задачи о блинах свидетельствует, что участие ЭВМ в оптимальном решении задачи обеспечивается, по крайней мере, двумя способами: предварительной записью в память или программу кодов оптимальных шагов решения, найденных путем полного перебора или эвристическими методами; программированием алгоритма формирования кодов оптимальных шагов в процессе решения задачи.

Первый способ, очевидно, пригоден лишь для задач с полной информацией, тогда как для принятия решений в условиях неопределенности приходится прибегать ко второму способу, более свойственному "думающей" машине. Эти же способы читатель может использовать и для решения с помощью карманной ЭВМ несложных "интеллектуальных" задач — переводов определенных текстов с иностранного языка, сочинений стихов из набора рифмующихся слов, составлений самообучающихся программ. Но наиболее интересной областью развития "интеллекта" микрокалькулятора являются игры.

В теории игр изучаются методы принятия решений обычно в конфликтных ситуациях, когда сталкиваются интересы нескольких (чаще всего двух) сторон. Не

следует думать, что главная задача разработчиков этой теории заключается в помощи преферансистам — конфликтные ситуации присущи всем жизненно важным сторонам человеческой деятельности.

Элементами игр являются ходы, разделяемые на личные, выбираемые игроком из допустимого множества, и случайные, не зависящие от воли играющего. Разыгрывая партию игры, стороны придерживаются определенных стратегий — наилучших, по их мнению, путей получения результата. Теоретически любая игра с двумя участниками представима в форме таблицы (матрицы) со строками, соответствующими возможным стратегиям первого игрока, и столбцами — второго. Элементы a_{ij} такой матрицы определяют выигрыш первого игрока при выборе им i -й стратегии и противником — j -й. Если выигрыш одного игрока равен проигрышу второго, то игру называют игрой с нулевой суммой.

Наибольший из наименьших элементов строк называют нижней чистой ценой (*максимином*) игры α , а наименьший из наибольших элементов столбцов — верхней чистой ценой (*минимаксом*) игры β . Всегда $\alpha \leq \beta$, так как выигрыш первого игрока не может превысить проигрыш второго игрока. Если максимин и минимакс расположены в одной клетке, то игра имеет седловую точку при значении $\alpha = \beta$, называемом чистой ценой игры и соответствующей оптимальному выбору стратегий обоим игрокам. Игру называют справедливой, если чистая цена игры равна нулю. Седловая точка существует не во всех играх, и в таких случаях, как мы увидим далее, оптимальное решение соответствует изменению стратегий в процессе игры.

Многие выводы теории игр применимы к ситуациям, в которых "противником" являются стихийные силы природы, и в таких "играх с природой" оптимальная стратегия зависит от статистических сведений о возможном поведении противника.

4.2. Игры с камешками

Вероятно, наиболее древними, судя по простоте "реквизита", являются игры двух или нескольких участников с множеством камешков или других предметов, поочередно добавляемых игроками в кучу или забираемых из нее согласно определенным правилам до выполнения условия, соответствующего выигрышу одного из участников. В подобных играх, состоящих только из личных ходов, теоретически всегда возможно найти наилучшую стратегию одного из участников, гарантирующую его выигрыш или, если это допустимо правилами игры, по крайней мере, ничью. Знание этой стратегии позволяет не упустить "законный" выигрыш и избежать "незаслуженного" проигрыша.

Математиков издавна интересовали поиски оптимальных стратегий в играх. Еще в 1612 г. в г. Лионе вышло в свет сочинение французского математика Баше де Мезиряка, в котором он среди других "забавных и приятных задач с числами" рассмотрел игру, вошедшую в литературу под названием игры Баше. На "языке камешков" ее можно описать следующим образом: два игрока поочередно кладут в одну кучу от одного до десяти камешков, причем выигрывает дополнивший эту кучу до 100 камешков.

Известны различные разновидности игры Баше, отличающиеся максимально допустимым числом N камешков, которые игрок может класть в кучу при очередном ходе, числом S_m камешков, определяющим окончание игры, и условием выигрыша — выигрывает или проигрывает положивший последний камешек. Возможна и обратная игра Баше, перед началом которой составляется куча из S_m камешков, из которой игроки поочередно забирают от одного до N камешков, причем выигрывает или проигрывает взявший последний камешек.

Предположим, что одним из участников этих игр явля-

ется представляющий Вас программируемый микрокалькулятор. Вам, как его "тренеру", необходимо составить программу, обеспечивающую выигрыш микрокалькулятора. Так как при каждом ходе в этой игре возможно N вариантов ее продолжения, а минимальное число ходов равно S_m/N , то число различных вариантов игры намного превышает $(S_m/N)^N$ и может оказаться чрезмерно большим для поиска оптимального решения методом полного перебора даже с помощью ЭВМ высокой производительности. Поэтому воспользуемся эвристическим методом разбиения задачи на подзадачи.

В разновидности прямой игры Баше с выигрышем игрока, кладущего в кучу последний камешек, заданное число S_m камешков соответствует выигрышной (особой) позиции $S_n = S_m$. Для ее гарантированного достижения нам необходимо предыдущим ходом занять особую позицию с числом $S_{n-1} = S_n - (N+1)$ камешков. Тогда при любом допустимом числе m_n камешков, добавленных противником последним ходом, он не сможет достичь особой позиции S_n , которую мы достигаем следующим ходом, добавив $(N+1) - m_n$ камешков. Рассматривая пути достижения особой позиции S_{n-1} , легко убедиться, что для этого следует занять предыдущую особую позицию: $S_{n-2} = S_{n-1} - (N+1)$.

Продолжая подобный анализ, несложно установить, что для выигрыша достаточно последовательно занимать особые позиции, отличающиеся на $N+1$ камешков, добавлением $N+1 - m_i$ камешков после каждого хода противника m_i камешками. Игрок, придерживающийся этой оптимальной стратегии, всегда выигрывает, если он начинает игру и первым ходом занимает особую позицию. Число камешков, которое необходимо klaсть начинаяющему игру для занятия особой позиции, очевидно, равно остатку от деления S_m на $N+1$. Если S_m кратно $N+1$, то исходная позиция особая (остаток от деления равен нулю) и начи-

нающий игру не сможет своим ходом достичь следующей особой позиции — в этом случае выигрывает второй игрок, придерживающийся оптимальной стратегии.

Последовательности возможных позиций в рассмотренной разновидности игры Баше при $N = 3$ для $S_m = 9$ и $S_m = 8$ (исходная позиция особая) показаны на рис. 42, *а* в виде графов, вершины которых, соответствующие особым позициям, обведены двойными кружками.

В разновидности игры Баше с проигрышем кладущего последний камешек последняя особая позиция определяется числом камешков $S_n = S_m - 1$, так как следующим ходом противник вынужден положить не менее одного камешка (рис. 42, *б*). В этом случае первая особая позиция S_1 равна остатку от деления $S_n = S_m - 1$ на $N + 1$, и при оп-

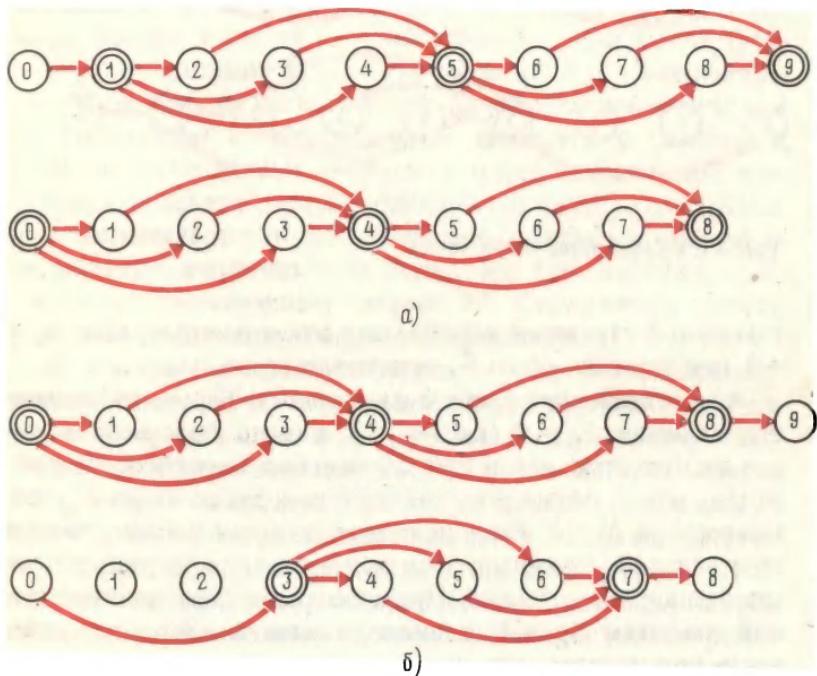


Рис. 42. Графы и игры Баше (*а*, *б*)

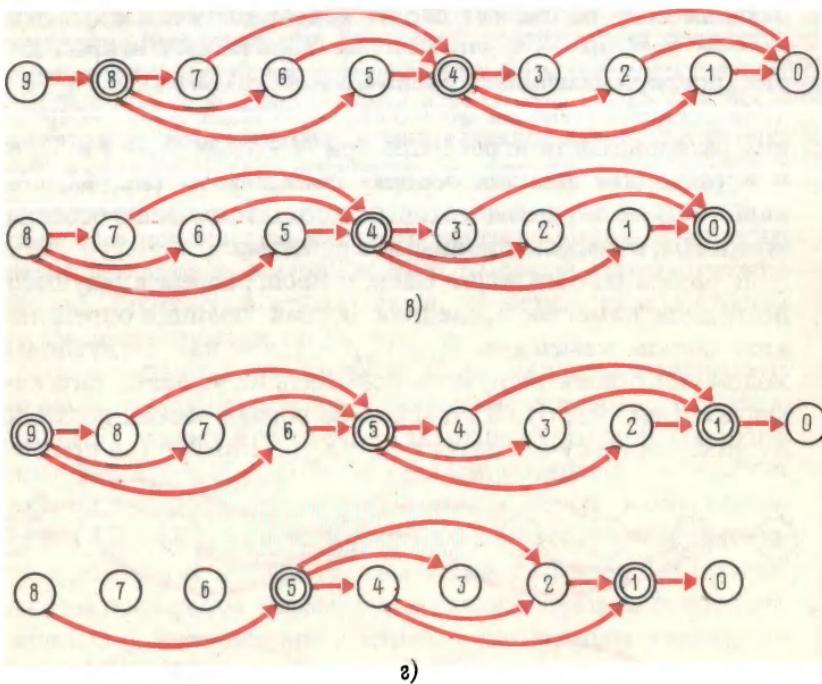


Рис. 42. Графы игры Баше (в, г)

тимальной стратегии выигрывает второй игрок, если $S_1 = 0$, или первый – если S_1 отлично от нуля.

В обратной игре Баше с выигрышем берущего последний камешек $S_n = 0$ (рис. 42, в), а число камешков, которое необходимо взять для достижения первой особой позиции, равно остатку от деления исходного числа S_m камешков на $N + 1$. Если исходная позиция особая, то при оптимальной стратегии выигрывает второй игрок, в противном случае – первый. При проигрыше берущего последний камешек $S_n = 1$, а число камешков, которые нужно взять при первом ходе для занятия особой позиции, равно остатку от деления $S_m - 1$ на $N + 1$ (рис. 42, г).

Успешное участие микрокалькулятора "Электроника Б3-21" в игре Баше при предоставлении первого хода противнику обеспечивает следующая программа.

Программа игры Баше

P5	C _x	P2	0	0	P4	F5	x ≠ 0	PC _x	↑	F7	-
x < 0	PC _x	C _x	P4	F2	÷	P2	↑	F8	XY	-	x ≠ 0
PC _x	↑	F7	-	x ≠ 0	F8	x < 0	F4	F2	+	P2	XY
P4	F8	-	x ≥ 0	PC _x	7	7	C/П	F4	C/П	P5	БП
0	F2	↑	F5	-	x ≠ 0	P/-/	x < 0	P-	БП	P/-/	

При условии выигрыша берущего из кучи последний камешек после ввода этой программы в регистры 7 и 8 соответственно заносят заданные числа $N + 1$ и S_m . Число камешков m_i , взятых противником после каждого его хода, вводят в регистр X и нажимают клавишу С/П (первый раз – клавиши В/О и С/П). Ответный ход микрокалькулятора в виде числа взятых камешков высвечивается на индикаторе и автоматически заносится в регистр 4, а общее число взятых камешков и предыдущий ход противника хранятся соответственно в регистрах 2 и 5. Если игра заканчивается проигрышем микрокалькулятора, то на индикаторе высвечивается нуль, но при выигрыше он "радостно" высвечивает цифры 77. Содержимое регистров 7 и 8 после каждой партии сохраняется.

В программе предусмотрена "проверка честности" противника – если он указывает недопустимое число камешков $m_i > N$ или $m_i = 0$, то микрокалькулятор "протестует", высвечивая цифры 00, но после исправления хода игру можно продолжить нажатием клавиши С/П. При обыкновенной исходной позиции и оптимальной стратегии противника микрокалькулятор проигрывает, но во всех остальных случаях он выигрывает, причем для введения противника в заблуждение микрокалькулятор разнообразит свои ходы, затрудняя разгадку своей стратегии.

Эта программа пригодна и для остальных вариантов

игры Баше, но в этом случае вместо S_m в регистр 8 перед игрой вводят число $S_m - 1$, если проигрывает берущий или кладущий последний камешек. Для игр Баше с первым ходом микрокалькулятора в программе следует заменить операторы $x \neq 0$ и PC_x первой строки операторами НОП, а для первого хода микрокалькулятора перед пуском программы первый раз в регистр X следует ввести 0. В этом случае "проверка честности" противника ограничивается лишь проверкой $m_i \leq N$.

Для игр Баше с обычновенной исходной позицией и первым ходом микрокалькулятора (или особой исходной позицией и вторым его ходом) пригодна очень простая программа:

P2 $x=0 \uparrow F4$ C/П F7 $\uparrow F2 - C/П$ БП Р0

Перед ее первым пуском в регистры 4 и 7 соответственно вводят число S_1 камешков, которые микрокалькулятор должен "взять" для занятия первой особой позиции, и $N + 1$, а в регистр X — нуль. Перед последующими пусками программы (нажатием только клавиши С/П) в регистр X заносят число камешков, взятых противником микрокалькулятора очередным ходом.

Несколько сложнее программирование игр с двумя множествами камешков или других предметов. На Востоке с глубокой древности известна игра Цзяньшицы ("Выбирание камней"), перед началом которой составляют две кучи камешков и игроки поочередно забирают либо одинаковое число камешков из обеих куч, либо произвольное число камешков из одной кучи, причем выигрывает взявший последний камешек.

Математический анализ показывает, что для этой игры также характерны особые позиции (a_k, b_k) с числом камешков a_k и b_k в кучах в k -й особой позиции. Связь между особыми позициями определяется следующими соотношениями: в нулевой особой позиции $a_0 = b_0 = 0$, в последующих особых позициях a_k равно наименьшему целому чис-

лу, отсутствующему в предыдущих особых позициях, и $b_k = a_k + k$. Если записывать особые позиции при $a_k < b_k$, то ближайшими к нулевой будут (1, 2), (3, 5), (4, 7), (6, 10), (8, 13) и т.д.

В этой игре, как и в игре Баше, из особой позиции можно перейти только в обыкновенную и наоборот. Поэтому оптимальная стратегия заключается в занятии особой позиции с последующим переходом после каждого хода противника на особую позицию с меньшим номером k до достижения выигрышной позиции (0, 0). Следовательно, если исходная позиция особая, то первый игрок (начинающий игру первым ходом) достигает лишь обыкновенной позиции, а его противник при оптимальной стратегии занимает особую позицию и в последующем выигрывает. Если исходная позиция обыкновенная, то при оптимальной стратегии всегда выигрывает первый игрок.

Для обучения микрокалькулятора оптимальной стратегии "Выбирания камней" целесообразно воспользоваться формулами [4] для числа камней в k -й особой позиции:

$$a_k = E[k(1+\sqrt{5})/2]; \quad b_k = E[k(3+\sqrt{5})/2],$$

где символом $E(x)$ обозначена целая часть содержимого скобок.

Заставив микрокалькулятор перебрать особые позиции с номерами $k = 0, 1, 2, \dots$ до такой, к которой возможен переход из заданной обыкновенной позиции, получим решение задачи на каждом ходе. Если заданная позиция особая, то микрокалькулятору придется очередным ходом занять обыкновенную позицию в ожидании будущих ошибок противника. Подобному подходу соответствует следующая программа.

Программа игры в "Выбирание камней"

C_x	P7	\uparrow	F8	X	P4	1	+	$x \geq 0$	P1	F4	-
\uparrow	F7	+	ПП	P,	F5	\uparrow	F6	ПП	P,	F7	1
+	B/O	P5	XY	P6	F2	-	\leftarrow	F6	\uparrow	F3	-

$$\begin{array}{ccccccccc} \uparrow & x \neq 0 & FC_x & \rightarrow & x \neq 0 & P+ & - & x \neq 0 & 8 \\ - & F7 & P6 & C_x & C/\Pi & x \geq 0 & 8 & B/O & \leftarrow x \geq 0 & 8 & B/O \end{array}$$

После ввода программы в регистр 8 заносят число $-(\sqrt{5}+1)/2$, а исходные числа камней a и b в кучах ($a < b$) – соответственно в регистры 2 и 3. Высвечивание нуля после каждого пуска программы нажатием клавиши B/O и C/P свидетельствует об окончании "размышлений" микрокалькулятора над своим ходом. Оставшиеся после этого хода числа камней в кучах хранятся соответственно в регистрах 5 и 6, что позволяет сравнить их с исходными числами камней (в начале игры или после предыдущего хода противника).

Если микрокалькулятору приходится ходить из особой позиции с номером k , он переводит ее в обыкновенную по правилу $a_i = a_k (=P5)$, $b_i = k (=P6)$. Это правило нарушается лишь при $a_k = 2 (=P2)$ и $b_k = 1 (=P3)$, когда микрокалькулятор в связи с неизбежным проигрышем "отказывается" ходить и ему засчитывается поражение. Впрочем из той же особой позиции, записанной $a_k = 1 = P2$, $b_k = 2 = P3$, он отвечает безнадежным ходом $1 = P5$, $1 = P6$, очевидно, в слабой надежде на грубую ошибку противника.

Авторы должны признаться, что обученный ими с помощью приведенной программы микрокалькулятор в первой же партии беззастенчиво (что и следовало ожидать от "бездушной" машины) обыграл их со следующими результатами:

1. $P2 = 100$, $P3 = 108 \rightarrow F5 = 12$, $F6 = 20$.
2. $P2 = 11$, $P3 = 19 \rightarrow F5 = 11$, $F6 = 18$.
3. $P2 = 10$, $P3 = 18 \rightarrow F5 = 10$, $F6 = 6$.
4. $P2 = 5$, $P3 = 10 \rightarrow F5 = 5$, $F6 = 3$.
5. $P2 = 3$, $P3 = 4 \rightarrow F5 = 1$, $F6 = 2$.
6. $P2 = 1$, $P3 = 1 \rightarrow F5 = 0$, $F6 = 0$.

Максимально допустимая начальная разность числа камней в кучах ограничена лишь точностью вычисления

содержимого регистра 8 и для восьмиразрядного микрокалькулятора может достигать 10^6 .

В рассмотренной игре при определении перехода от заданной позиции (a, b) , где $a < b$, неоценимую помощь Вам может оказать и непрограммируемый микрокалькулятор. В этом случае следует предварительно вычислить числа $B_1 = (\sqrt{5} + 1) : 2 = 1,618\dots$ и $B_2 = (\sqrt{5} + 3) : 2 = 2,618\dots$ и воспользоваться следующим алгоритмом:

1. Если $aB_2/B_1 > b$, то $k = b - a$, $a_k = E[kB_1]$, $b_k = a_k + k$, иначе перейти к п.2.

2. Вычислить $k_1 = E[(a + 1)/B_1]$, $k_2 = E[(a + 1)/B_2]$, $a_{k_1} = E[k_1 B_1]$, $b_{k_2} = E[k_2 B_2]$.

3. Если $a_{k_1} = a$, то $a_k = a$, $b_k = a_k + k_1$, иначе $b_k = b_{k_2} = a$ и $a_k = a - k_2$.

Например, для заданной позиции (42,78) согласно этому алгоритму вычисляем $1,618 \cdot 42 = 67,956 < 78$, $k_1 = 26$; $k_2 = 17$, $a_{k_1} = 42$ и, следовательно, $a_k = 42$, $b_k = 42 + 26 = 68$.

В Европе давно известна еще более сложная игра с тремя кучами предметов, которая в 1901 г. получила название Ним. В этой игре партнеры поочередно забирают из одной выбранной кучи произвольное число предметов, причем в зависимости от предварительной договоренности выигрывает или проигрывает взявший последний предмет. Особую популярность приобрела эта игра в эпоху становления кибернетики. По свидетельству известного ученого при демонстрации автомата для игры в Ним на западноберлинской ярмарке в 1951 г. посетители игнорировали даже бар, а для наведения порядка пришлось вызывать полицию. Другой специалист-кибернетик отмечал, что программисты "продирают игрой Ним голос каждой новорожденной ЭВМ".

Теория оптимальной игры в Ним также основана на занятии особых позиций: игрок, начинающий с обычной позиции и с каждым ходом занимающий особую позицию, в конечном итоге забирает последний (или пред-

последний) предмет и выигрывает. Для определенности будем полагать, что выигрывает взявший последний предмет, и обозначим позиции последовательностями (a, b, c) чисел предметов в кучах. Согласно теории игры, тип позиции определяется суммой двоичных кодов этих чисел: если каждый разряд такой суммы содержит числа 0 и 2, то позиция особая [1].

В одном из классических вариантов игры Ним исходной является позиция (3, 4, 5). В приведенной ниже программе исходная позиция может отличаться от классической только первым числом.

Программа классического варианта игры Ним

F8	C/P	P3	F6	ПП	P5	F6	ПП	4	F6	ПП	4
F6	ПП	4	5	7	ПП	4	9	ПП	4	1	3
ПП	4	9	↑	F2	+	P2	↑	F3	—	$x < 0$	8
5	+	$x < 0$	P9	F7	+	$x \neq 0$	P9	1	0	—	$x \neq 0$
P9	$x < 0$	PC _x	0	0	B/O	F3	↑	F2	C/P	БП	P↑

После ввода программы в регистры 6,7 и 8 следует занести соответственно числа 11,100 и 145. Введя в регистр X код исходной позиции (число 345 или любое другое, отличающееся от него первой цифрой, большей единицы) и нажав клавиши B/O и C/P, противник увидит на индикаторе ответный ход микрокалькулятора – код позиции 145. Следовательно, микрокалькулятор “берет” из первой кучи количество предметов, необходимое для перехода в особую позицию.

В дальнейшем противник микрокалькулятора, вводя в регистр X код занимаемой им позиции, нажимает только клавишу C/P. Микрокалькулятор, затратив на “обдумывание” несколько десятков секунд, высвечивает на индикаторе ответный код занимаемой им особой позиции. При этом код предыдущей позиции противника сохраняется в регистрах 3 и Y. Высвечивание двузначного числа

означает, что из первой кучи все предметы забраны. Если противник в предчувствии своего проигрыша не "берет" ни одного предмета, то микрокалькулятор отвечает тем же. В остальных случаях окончание игры и свою победу микрокалькулятор отмечает высвечиванием цифр 00. К этому же результату приводит и большинство неправильных ходов противника, пытающегося "обмануть" микрокалькулятор.

При игре Ним микрокалькулятор может выступать и в роли весьма квалифицированного советника. В этом случае при числе предметов в каждой куче не более 63 микрокалькулятору достаточно запомнить следующую программу.

Программа рекомендаций игры Ним

F2	ПП	5	F3	ПП	5	3	2	P8	C _x	P4	C _x
→	1	0	—	x=0	÷	F8	↑	F4	+	P4	F8
2	÷	P8	1	—	x<0	XY	F4	C/П	P4	6	4
P8	F4	↑	F8	—	x≥0	7	P4	1	0	↑	F5
+	→	P5	F8	2	÷	P8	1	—	x<0	F6	B/O

Перед первым пуском этой программы (для каждого пуска нажимают клавиши B/O и C/П) стек памяти очищают. Два из трех чисел заданной позиции (a, b, c) вводят в регистры 2 и 3, а выполнение программы приводит к высвечиванию третьего числа соответствующей особой позиции. Возможны следующие варианты "советов":

третье вычисленное число равно исходному — заданная позиция особенная и Вам следует перейти в любую допустимую позицию в надежде на незнание противником оптимальной стратегии;

вычисленное число меньше исходного — прислушайтесь к "совету" микрокалькулятора и возьмите из соответствующей кучи надлежащее число предметов;

вычисленное число больше исходного — введите другие пары чисел из заданной позиции до получения одного из предыдущих "советов".

Например, при заданной позиции (12, 17, 25) ввод 12 = P2, 17 = P3 приведет к вычислению $29 > 35$. Ввод 12 = P2, 25 = P3 также приведет к неудовлетворительному результату $21 > 17$, но ввод 17 = P2, 25 = P3 даст число $8 < 12$. Это означает, что из кучи, содержащей 12 предметов, следует взять $12 - 8 = 4$ предмета, что обеспечит переход в особую позицию (8, 17, 25).

В некоторых играх с изменением игроками числа предметов выигрыш определяется некоторым дополнительным условием. В известной игре подобного рода исходным является нечетное число спичек в куче, из которой игроки по очереди берут от одной до четырех спичек, причем выигрывает набравший их четное число. Участие микрокалькулятора в этой игре обеспечивается занесением в его память следующей программы.

Программа игры со спичками

P8	C _x	P2	P3	/ - /	4	+	$x \geq 0$	9	F8	+	$x \geq 0$
9	P8	$x \neq 0$	P-	F3	-	/ - /	P3	2	-	$x < 0$	P \div
1	+ $x \neq 0$	P4	\uparrow	F8	+	6	- $x < 0$	F5	XY		
5	- $x \geq 0$	P7	4	\uparrow	F2	+	P2	F8	- $x \geq 0$		
P9	/ - /	\uparrow	F2	+	C/P	/ - /	P8	XY	C/P	BП	F \uparrow

Противник вводит в регистр X нечетное начальное число S спичек и нажимает клавиши B/O и C/P. "Поразмыслив", микрокалькулятор высвечивает число "взятых" им спичек (число оставшихся спичек хранится в регистре Y). В последующем противник вводит в регистр X также число взятых им спичек и нажимает только клавишу C/P. Игра оканчивается, когда микрокалькулятор, "забрав" последние спички или "убедившись", что это сделал противник, высвечивает сумму всех своих спичек.

Если противник взял более четырех спичек, то микрокалькулятор выразит свое "возмущение" высвечиванием числа лишних спичек с отрицательным знаком и противнику придется повторить ход с соблюдением

правил. Если противник не берет ни одной спички, то микрокалькулятор отвечает тем же, заставляя противника играть по правилам. Высвечивает отрицательное число микрокалькулятор и в том случае, когда противник берет больше спичек, чем оставалось в куче. Наконец, микрокалькулятор отказывается играть с особой позиции ($S = 6n + 1$), которая обеспечивает выигрыш противника при его оптимальной стратегии.

В некоторых играх изменяют распределение предметов в кучах. Примером может служить игра Гранди, в которой игрок делит исходную кучу из S предметов на две *неравные* части и в дальнейшем игроки по очереди делят таким же образом одну из куч до тех пор, пока не образуются кучи из *одного или двух предметов*. Игрок, который очередным ходом достигает этого состояния куч, выигрывает. Оптимальный алгоритм этой игры несложно составить методом полного перебора. Поэтому приведем лишь программу для $S = 10$ и первого хода противника микрокалькулятора.

Программа игры Гранди

\uparrow	8	2	—	$x=0$	X Y	7	2	1	БП	R X	6
3	1	C/P	\uparrow	5	3	1	1	—	$x=0$	P5	3
3	2	1	1	BП	/—	4	2	2	1	1	C/P
C_x	2	2	2	1	1	1	1	\uparrow	0	0	C/P
БП	P0										

Противник первым ходом заносит в регистр X последовательность 91, 82, 73 или 64 чисел предметов в кучах, на которые он делит исходную кучу, и нажимает клавиши В/О и С/П. После трехзначного ответа микрокалькулятора, соответствующего числам предметов в трех кучах, противник вводит последовательность из четырех чисел (начиная с больших) и нажимает, как и при последующих ходах, только клавишу С/П. Свой выигрыш микрокалькулятор отмечает высвечиванием цифр 00, а достигнутую

им выигрышную позицию можно вызвать нажатием клавиши XY. Повторные партии можно начинать, нажимая только клавишу С/П.

А теперь, читатель, попытайтесь решить следующие задачи:

1. Маша и Саша по очереди обрывают один или два лепестка у цветка с 13 лепестками, условившись, что выигрывает сорвавший последний лепесток. Как поступать Маше, начинающей игру?

* Ответ. Согласно оптимальной стратегии игры Баше Маше следует сорвать один листок, а затем дополнять ходы Саши (срывать два лепестка, если он сорвал один, и наоборот).

2. Составьте простейшую программу игры Баше для начинающей игру вторым при особой исходной позиции и выигрыше взявшего последний предмет.

Ответ. Записав в регистр 7 число $N + 1$, составьте программу ↑ F7 – /-/ С/П БП Р0.

3. Восстановите алгоритм игры Ним с исходной позицией (17, 13, 9) взятием не более трех предметов из одной кучи за один ход и выигрышем взявшего последний предмет по следующей программе (17 = Р2, 13 = Р3, 0 = Р4) для микрокалькулятора, вступающего в игру вторым.

cos	x < 0	F7	4	X	sin x ≥ 0	→	F4	4	-	P4	
x < 0	F4	0	P4	F2	↑	F8	-	x ≥ 0	8	P2	3
P8	C/П	F3	4	-	P3	x < 0	F4	0	P3	F4	↑
F8	-	x ≥ 0	Fx	P4	БП	÷	F2	4	-	P2	x < 0
F4	0	P2	F3	↑	F8	-	x ≥ 0	F/-/	P3	БП	÷

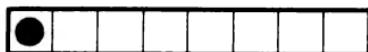
Противник вводит в регистр X номер 1, 2 или 3 кучи и, после запятой, число взятых из нее предметов, нажимая каждый раз клавиши В/О и С/П. Если микрокалькулятор ответным ходом "берет" предметы из той же кучи, то на индикаторе высвечивается число оставшихся в ней пред-

метов, если из другой — на индикатор выводится число 3 (количество оставшихся предметов в каждой куче определяется содержимым регистров 2, 3 и 4).

4.3. Микрокалькулятор играет в шахматы

Участники большинства настольных игр по очереди изменяют по установленным правилам расположение фигур (предметов или символов) на доске или таблице определенной формы. Простейшим примером может служить игра на прямоугольной доске (рис. 43, а) со следующими правилами: игроки по очереди передвигают фишку вправо на несколько (от одной до N) клеток, причем выигрывает (или проигрывает) игрок, первым передвинувший фишку на крайнюю правую клетку. Какова оптимальная стратегия такой игры? На этот вопрос легко ответить, заметив ее аналогию с игрой Баше. Следовательно, участником этой игры может быть и микрокалькулятор с программой игры Баше.

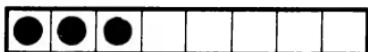
Подобные игры с двумя (рис. 43, б) или тремя (рис. 43, в) фишками, расположенными в исходном положении на



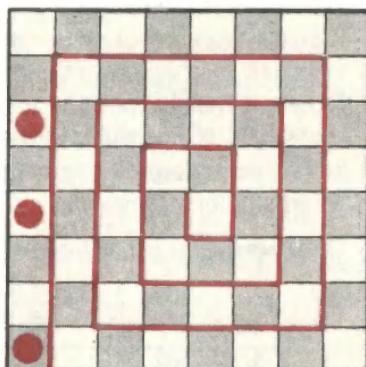
а)



б)



в)



г)

Рис. 43. Игры на прямоугольных досках

доске и передвигаемыми по правилам, аналогичным правилам взятия предметов в играх Цзяньшицы и Ним, отличаются и аналогичными оптимальными стратегиями. Для них также можно использовать ранее приведенные программы. Рассматриваемые игры не обязательно связанны с досками, подобными изображенным на рис. 43, а – в. Так как существенна лишь последовательность мест, занимаемых фишками, то для них, в частности, пригодна и шахматная доска, на которой можно выделить спиральную (рис. 43, г) или иную последовательность клеток.

Кстати, шахматная доска часто имеется "под руками" и ее удобно использовать и для многих других игр. Для примера рассмотрим игру, называемую "Однокая пешка", в которой исходному положению соответствует пешка на клетке $h\ 8$ в правом верхнем углу шахматной доски. Игроки по очереди передвигают пешку на соседнюю левую или нижнюю клетку или по диагонали налево вниз (рис. 44). Выигрывает игрок, первый передвинувший пешку на клетку $a\ 1$.

Методом разбиения задачи на подзадачи несложно установить, что для выигрыша следует занимать особые позиции, отмеченные на доске крестиками. Обозначим клетки кодом xy , где x и y – соответственно номера вертикалей и горизонталей доски. Тогда исходной позиции будет соответствовать код $xy = 88$, а выигрышной – код $xy = 11$. В соответствии с правилами игры можно составить следующую программу для микрокалькулятора, начинаящего первым.

Программа игры "Однокая пешка"

P2	1	0	-	$x < 0$	F0	F2	XY	P3	-	1	0
÷	ПП	F,	1	0	X	P2	F3	ПП	F,	↑	F2
+	C/П	БП	P0	↑	π	X	cos	$x \geq 0$	P6	-	↑
XY	B/O										

Перед каждым пуском программы (первый пуск – кла-

вишами В/О и С/П, последующие — только клавишей С/П) в регистр X вводят код xy положения пешки в начале игры или после хода противника. После выполнения программы на индикаторе высвечивается код xy клетки, на которую "ставит" пешку микрокалькулятор ответным ходом.

Можно заметить, что условия этой игры отличаются от Цзяньшидзы лишь тем, что числа x и y можно изменять только на единицу, причем допускается одновременное уменьшение одного из них и увеличение другого. Полная аналогия с игрой Цзяньшидзы (при ограничении числом 8 исходных значений x и y) характерна для игры, называемой "Одинокий ферзь". В исходном положении этой игры ферзь располагают на любой клетке верхнего правого квадрата шахматной доски, размером 3×3 клетки (рис. 45). Игра по очереди ходят ферзем влево, вниз или по диагонали налево вниз на любое число клеток, допускаемое границами доски. Причем выигрывает игрок, занявший клетку $a1$ или в нашем коде $xy = 11$.

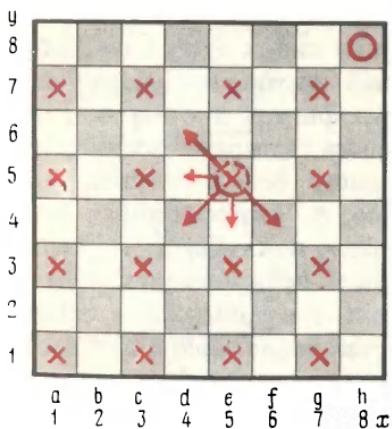


Рис. 44. Одинокая пешка

13*

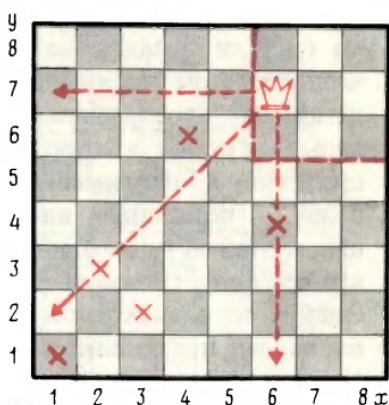


Рис. 45. Одинокий ферзь

Для успешного участия микрокалькулятора в этой игре с первым его ходом можно использовать следующую программу.

Программа игры "Одинокий ферзь"

P2	1	1	ПП	F4	6	4	ПП	P,	4	6	ПП
P,	3	2	ПП	P,	2	3	ПП	P,	F3	C/П	БП
P0	P4	P3	-	P5	$x \geq 0$	F-	\uparrow	F4	-	$x \neq 0$	3
$x < 0$	F5	F5	1	0	-	$x \geq 0$	3	1	0	-	$x < 0$
PC _x	XY	$x \neq 0$	3	F2	B/O						

После ввода программы код $ху$ исходного положения ферзя заносят в регистр X и нажимают клавиши В/О и С/П. После выполнения программы на индикаторе высвечивается код $ху$ ответного хода. В последующем противник вводит в регистр X код $ху$ занимаемой им клетки и нажимает только клавишу С/П. Игра оканчивается выигрышем микрокалькулятора, высвечивающего код $ху = 11$.

Рассмотрим еще одну игру на шахматной доске, в начале которой три белые и три черные фигуры ("мушкетеры") располагаются, как показано на рис. 46, а. Игроки по очереди передвигают одну из своих фигур на любое число клеток соответствующей вертикали вперед или назад, не "перепрыгивая" через фигуру противника. Выигрывает игрок, которому удалось "запереть" все фигуры противника, например выигрыш белых соответствует позиция, показанная на рис. 46, б. Анализ особых позиций в этой игре методом подзадач показывает, что белые выигрывают, если они "запирают" первым ходом черную фигуру на ее исходной позиции, а в дальнейшем отвечают на ход противника вперед таким же ходом по другой диагонали и на каждый ход назад — продвижением своей фигуры по той же диагонали на число клеток, равное числу клеток, разделяющих фигуры на другой диагонали. При игре с микрокалькулятором в его програм-

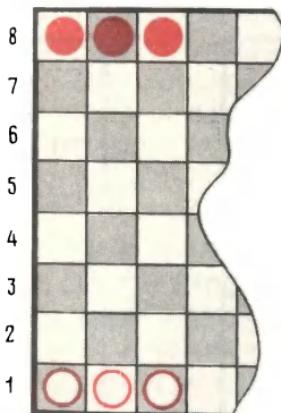
мную память следует ввести следующую программу
(микрокалькулятор "играет" белыми).

Программа игры в "Мушкетеры"

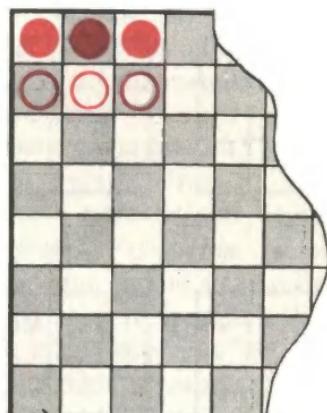
1	8	P2	2	8	P3	1	1	P4	2	1	P5
3	7	↑	C/П	↑	2	0	—	x<0	FБП	F2	XY
P2	—	↑	x≥0	F/-	F5	+	P5	БП	2	F4	XY
—	P4	БП	2	F3	XY	P3	—	↑	x≥0	F-	F4
+	P4	БП	2	F5	XY	—	P5	БП	2		

После ввода программы очищают регистр X и нажимают клавиши В/О и С/П. После выполнения программы на индикаторе высвечивается код первого хода микрокалькулятора $ху = 37$. После ввода в регистр X кода $ху$ клетки, занимаемой ходом противника, нажимают только клавишу С/П и микрокалькулятор отвечает высвечиванием кода $ху$ своего очередного хода.

Шахматную доску можно использовать, отделив квадрат размером 3×3 клетки, и для популярной игры, извест-



а)



б)

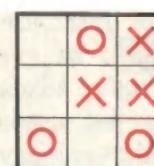
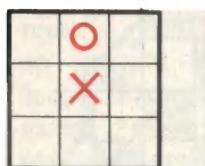
Рис. 46. Мушкетеры

ной нашим школьникам "крестики-нолики". В этой игре партнеры по очереди занимают клетки прямоугольной таблицы размером 3×3, причем выигрывает занявший первым три клетки по вертикали, горизонтали или диагонали. При оптимальной стратегии игрок, занимающий первым ходом центральную клетку, выигрывает, если противник ответным ходом занимает не угловую клетку (рис. 47, а), и сводит игру в ничью, если противник своим первым ходом занимает угловую клетку (рис. 47, б).

Оптимальный алгоритм этой игры для микрокалькулятора, начинаящего первый ход с занятия центральной клетки 9 (рис. 47, в), реализуется следующей программой.

Программа игры "Крестики-нолики"

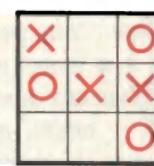
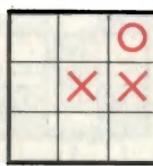
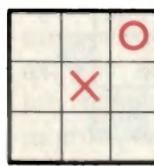
9	\uparrow	C/P	P/P	P8	P2	C/P	P/P	/-/	$x < 0$	F2	F2
P7		P/P	P8	C/P	P/P	/-/	$x < 0$	4	P/P	P8	C/P
/-/	0	C/P	F8	5	P/P	P-	F8	7	7	C/P	P7
F8	4	P/P	P-	F7	-	$x = 0$	F5	π	X	cos	B/O
P7	1	-	$x \neq 0$	F9	$x < 0$	9	8	+	\uparrow	P8	B/O



1	2	3
8	9	4
7	6	5

а)

б)



6	1	8
7	5	3
2	9	4

в)

г)

Рис. 47. Крестики-нолики

Игра начинается нажатием клавиши В/О и С/П, после чего на индикаторе высвечивается номер 9 клетки, занимаемой микрокалькулятором первым ходом. В последующем номер клетки, занимаемой противником, вводят в регистр X и нажимают только клавишу С/П. Выполнение программы каждый раз оканчивается высвечиванием номера клетки, занимаемой микрокалькулятором, а в случае его выигрыша или ничейного результата – соответственно цифр 77 или 0 (при выигрыше номер занятой микрокалькулятором клетки заносится в регистр Y и его можно вызвать нажатием клавиши $X \neq Y$).

Можно попытаться составить оптимальный алгоритм игры и при других вариантах первого хода, попытавшись использовать другие способы кодирования ходов, например кодами xy , соответствующими номерам вертикалей и горизонталей, или номерами клеток "магического квадрата" (рис. 47, г), у которого сумма номеров по диагоналям вертикалям и горизонталям равна 15.

"Крестики-нолики", как и ранее рассмотренные игры, имеют множество аналогов, часто совершенно непохожих по форме, но характеризующихся одинаковой оптимальной стратегией при одинаково сформулированных условиях игры. В одном из таких аналогов [2] партнеры по очереди берут по одной из карточек с надписями "рыба", "клип", "нить", "небо", "сон", "бусы", "тор", "сеть", "река", причем выигрывает взявший первым три карточки с одинаковыми буквами в словах. Разместив предварительно эти карточки в три ряда так, чтобы карточки с одинаковой буквой располагались по горизонталям, вертикалям и диагоналям, можно пронумеровать их согласно рис. 47, в и использовать для этой игры программу для "крестиков-ноликов".

В другой игре партнеры по очереди берут одну игральную карту от туза до девятки, которым соответственно присвоены номера от 1 до 9. Выигрывает набравший

первым три карты с суммарной "стоимостью" 15. Эта игра аналогична "крестики-ноликам" при нумерации клеток, показанной на рис. 47, 2, и для участия в ней микрокалькулятора достаточно лишь соответственно изменить "стоимость" карт.

Участники игры "Города и дороги" по очереди занимают (закрашивая своим цветом) по одной из обозначенных буквами дорог (рис. 48), причем выигрывает закрасивший первым все дороги, ведущие к одному из городов. Если заменить буквенные обозначения дорог числами, указанными в скобках, то в этой игре также может принять участие микрокалькулятор с программой для игры в "крестики-нолики".

К этим играм близки и многие другие, например игра "Тригекс" [4], участники которой по очереди занимают вершины фигуры, изображенной на рис. 49. Выигрывает

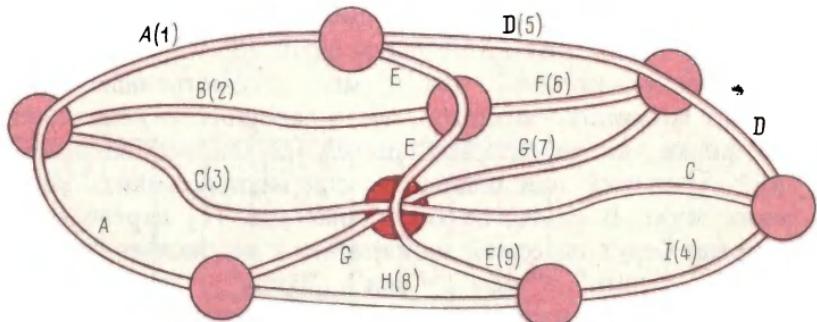


Рис. 48. Города и дороги

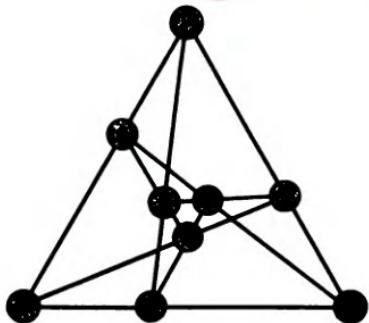


Рис. 49. Тригекс.

занявший первым три вершины, лежащие на одной прямой. Микрокалькулятор "Электроника Б3-21" достигает успеха в этой игре, запомнив следующую программу и делая первый ход.

Программа игры "Тригекс"

1	C/P	↑	8	-	x ≥ 0	F2	4	C/P	↑	3	BП
P9	2	+	x ≥ 0	R÷	2	БП	RXY	1	+	x = 0	P5
4	C/P	↑	9	БП	P9	1	+	x = 0	FВП	9	C/P
↑	5	БП	P9	1	+	x = 0	R-	9	C/P	↑	4
БП	P9	7	C/P	↑	5	↑	7	7	C/P	БП	P0

Перед началом игры очищают регистр X и нажимают клавиши В/О и С/П, после чего микрокалькулятор высвечивает номер 1 вершины, занимаемой им первым ходом. В последующем противник вводит в регистр X номер занимаемой им вершины и нажимает клавишу С/П, что приводит к высвечиванию ответного хода микрокалькулятора. После двух ходов противника микрокалькулятор выводит на индикатор число 77, означающее, что противник проиграл или проиграет (ход микрокалькулятора в регистре Y) при любом следующем ходе.

Иногда кажущееся усложнение правил игры существенно упрощает выбор оптимальной стратегии. Если, например, в "крестиках-ноликах" разрешить обоим игрокам использовать символы обоих типов, то всегда выигрывает первый игрок, занимающий центральную клетку и в дальнейшем использующий для обозначения занимаемых клеток символы противника. При первом ходе противника в угловую клетку первый игрок в наихудшем случае выигрывает третьим ходом, в противном варианте — четвертым.

Подобная стратегия оптимальна и в других играх с симметрией допустимых положений. Простейшим примером может служить легко программируемая игра, в которой игроки по очереди кладут одинаковые круглые предметы

на круглый стол. Если первый игрок занял центр стола, то в дальнейшем ему достаточно "симметрично" повторять ходы противника. Однако возможность использования симметрии не всегда очевидна. Для примера рассмотрим игру с изображенной на рис. 50, а доской, в клетки которой игроки по очереди ставят свои фишki, причем клетку с номером 0 разрешается занимать только заключительным ходом. Проигрывает игрок, три фишki которого образуют вместе с заштрихованной клеткой четырехугольник, подобный показанным на рис. 50, а.

Для определения оптимальной стратегии в этой игре обратимся к игре Сим [2], в которой игроки по очереди закрашивают своим цветом ветви графа с шестью вершинами (рис. 50, б), причем проигрывает первым построивший треугольник из ветвей своего цвета. Если правила этой игры дополнить условием закрашивания последней ветви, соединяющей две противоположные вершины (например 1 и 4), то остальные ветви образуют симметричные пары. В этом случае второй игрок, симметрично повторяющий ходы первого, всегда выигрывает, так как первый игрок неизбежно вынужден первым построить "роковой" треугольник. Полным аналогом такой модифицированной игры Сим и будет наша игра, которую для определенности назовем "Асим".

Обозначив симметричные ветви графа и соответствующие клетки доски (рис. 50) такими номерами, чтобы их суммы равнялись 15, легко составить для игры Асим весьма простую программу для микрокалькулятора "Электроника Б3-21", начинаящего игру вторым:

↑ 1 5 – С/П БП Р0.

Таким образом, незначительное изменение правил игры позволило найти простую оптимальную стратегию, хотя поиск такой стратегии для игры Сим весьма сложен даже при использовании "больших" ЭВМ [2].

Очень заманчиво использовать шахматную доску и по

прямому назначению. Многие читатели, наблюдавшие за телевизионными передачами, посвященными чемпионату мира по шахматам в Маниле, видели А. Карпова, игравшего в часы отдыха в шахматы с... микро-ЭВМ. Емкость запоминающих устройств массовых микрокалькуляторов чрезмерно мала для программирования игры не только в шахматы, но и в шашки, но упрощенные варианты этих игр можно попытаться программировать и на микрокалькуляторах.

Известны, например "минишашки", в которые играют по обычным правилам, но на доске размером 4×4 клетки. Обозначая кодом xu клетки, занимаемые шашками, попытаемся обеспечить участие микрокалькулятора в "миниподдавках" (выигрывает игрок, потерявший все шашки) с помощью следующей программы.

Программа игры "Миниподдавки"

```

2      2      ↑      F4      –      x≠0 XY      F5      –      x≠0 P/-/   3
3      P8      C/Π    4      4      ↑      F4      –      x≠0 P,      F5      –
x=0    F9      P8      3      3      P7      БП      F–      F5      4      2      –
x=0    F9      3      3      P7      C/Π    F5      2      4      –      x=0 F9
P7      3      3      P8      7      7      C/Π    0      0      C/Π

```

В этой программе положение белых шашек (рис. 51) ото-

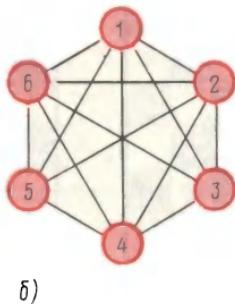
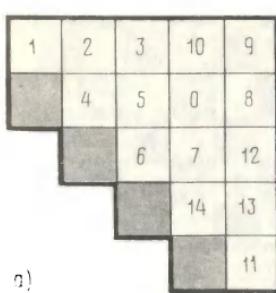


Рис. 50. Асим и Сим

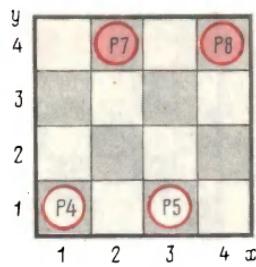


Рис. 51. Минишашки

бражается кодами xu в регистрах 4 и 5, а черных – в регистрах 7 и 8. Перед началом игры вводят в память исходную позицию $11 = P4$, $31 = P5$, $24 = P7$, $44 = P8$ и противник микрокалькулятора, вводя в регистр 4 или 5 код xu занимаемой им клетки, очищает регистр X и нажимает клавиши В/О и С/П. Микрокалькулятор отвечает высвечиванием на индикаторе и занесением в регистр 7 или 8 кода xu занимаемой им клетки. В последующем противник, занося код xu занимаемой клетки в регистр 4 или 5, соответствующий белой пешки, нажимает клавишу С/П. При выигрыше микрокалькулятора высвечиваются цифры 77, при попытке его обмана – цифры 00 ("противник дисквалифицирован").

А теперь перейдем к шахматам. В одной из древних разновидностей, известной под названием "магараджа", белые фигуры ходят по обычным правилам, но достигшие края доски пешки не заменяют другими фигурами. Черные имеют только одну фигуру (магараджу) с правами ферзя и коня, располагаемую перед началом игры на любой свободной клетке (рис. 52, а). Белые выигрывают,

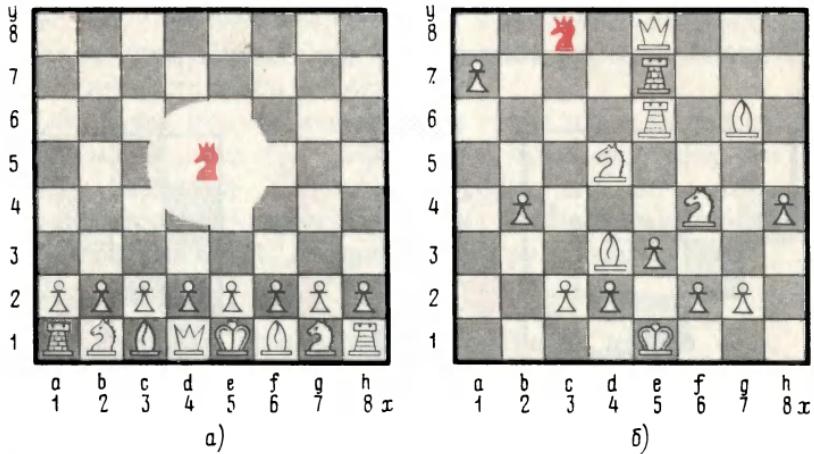


Рис. 52. Магараджа

сняв магараджу, черные – поставив мат белому королю. Боевые качества магараджи весьма высоки и эта игра протекает очень остро. Однако невозможность для черных жертвовать фигурами ставит их в неравное положение и справедливо ограничить игру 25 ходами. Легко ли при этом условии победить магараджу, читатель может проверить на собственной шахматной доске.

В книге [1] приведен "наиболее эффективный план победной кампании над магараджей", содержащий не более 23 ходов, ставящих магараджу в безвыходное положение. Такой план несложно запрограммировать на микрокалькуляторе с большей емкостью программной памяти, чем у микрокалькулятора "Электроника Б3-21", и для последнего необходимо составить еще более эффективный план. Положения фигур на доску закодируем следующим образом – обозначим кодом xu клетку шахматной доски, занимаемую белой пешкой, где x и y – соответственно номера вертикали и горизонтали (рис. 52), а поле, занимаемое другой белой фигурой, – кодом zxy , где z – порядковый номер соответствующей фигуры в исходном положении. Это позволит нам обмениваться с микрокалькулятором информацией в ходе игры с ним.

Предположим следующий план оптимальной игры микрокалькулятора, играющего белыми фигурами против магараджи:

1. $e2-e3$ 2. $Kg1-h3$ 3. $Kh3-f4$ 4. $Cf1-d3$ 5. $\Phi d1-h5$
6. $Kb1-c3$ 7. $a2-a4$ 8. $a4-a5$ 9. $a5-a6$ 10. $h2-h4$ 11. $Lh1-h3$ 12. $Lh3-g3$ 13. $Kc3-d5$ 14. $b2-b4$ 15. $Cc1-b2$ 16. $Lg3-g7$ 17. $a6-a7$ 18. $La1-a6$ 19. $Lg7-e7$ 20. $La6-e6$. Если магараджа занимает поле $f8$ или $g8$, то 21. $Cb2-g7$
22. $\Phi h5-e8$, иначе 21. $\Phi h5-e8$.

Положение фигур после заключительного (21 или 22-го) хода белых ферзем показано на рис. 52, б. Порядок ходов микрокалькулятора белыми фигурами

рами при их кодировании рассмотренным способом реализуется следующей программой.

Программа игры "Магараджа"

←	C/P	↑	1	4	C/P	↑	1	5	C/P	↑	1
6	C/P	↑	8	4	C/P	F2	C/P	F3	C/P	F4	C/P
↑	2	4	C/P	F5	C/P	F6	C/P	↑	1	7	C/P
F7	C/P	F8	C/P	↑	1	5	6	C/P	↑	5	8
-	x≥0	PC _x	3	7	7	C/P	↑	4	5	8	C/P

Перед началом игры в память микрокалькулятора вводят исходные данные $883 = P_2$, $873 = P_3$, $245 = P_4$, $322 = P_5$, $877 = P_6$, $116 = P_7$, $857 = P_8$, $53 = C_1$, $783 = C_2$, $764 = C_3$, $643 = C_4$, $485 = C_5$, $233 = C_6$. Для первых шести ходов микрокалькулятора (до высвечивания кода 233, соответствующего ходу белых 6. Кб1—с3) программу пускают нажатием клавиши В/О и С/П. Для последующих ходов в регистр X вводят код $ху$ клетки, занимаемой магараджей, и нажимают только клавишу С/П. После выполнения программы на индикаторе высвечивается код $ху$ или $zху$ кода микрокалькулятора.

С приведенной программой Ваш микрокалькулятор становится чемпионом мира по шахматам — в варианте "магараджа". Если Вы еще не приобрели программируемый микрокалькулятор, то не огорчайтесь — приведенный алгоритм обеспечит Вам победу над "магараджей" после его приобретения.

Рассмотренные игры отличаются общей особенностью — если обоим игрокам известна оптимальная стратегия и они не делают ошибок, то исход игры точно предсказуем после того, как станет известным, кто ходит первым. Эта особенность игр с полной информацией точно отображена в "кибернетическом" анекдоте об игре в шахматы между ЭВМ — после первого хода одной из них, вторая, перебрав все варианты, отвечает "Сдаюсь!". Анекдотичным в этой истории можно считать лишь то обстоятельство,

что, хотя шахматы формально относят к играм с полной информацией, ее объем настолько велик, что ЭВМ не может полностью ее использовать.

4.4. "Коктейль" из стратегий

Если среди возможных стратегий, которыми может руководствоваться игрок, есть такие, которые при любых ответных кодах противника лучше остальных стратегий, то их называют доминирующими. Естественно, что оптимальную стратегию следует искать среди доминирующих, игнорируя остальные, заведомо худшие. В рассмотренных нами играх для одной из сторон нам удавалось находить единственную доминирующую стратегию (с учетом равнозначных ходов в некоторых играх), гарантирующую выигрыш. Однако во многих играх, даже не содержащих случайных ходов, такую стратегию найти невозможно.

Рассмотрим игру "Чет-нечет", в которой один из соперников задумывает число 0 или 1, а второй пытается его отгадать, выигрывая один балл при угадывании и проигрывая один балл при "промахе". Матрица такой игры содержит элементы $a_{11} = 1$, $a_{12} = -1$, $a_{21} = -1$, $a_{22} = 1$, но верхняя (1) и нижняя (-1) цены игры не равны, седловая точка отсутствует и каждому из игроков опасно использовать единственную стратегию, так как противник воспользуется этим для выигрыша.

В подобных играх без седловой точки используют смешанную стратегию, состоящую в случайному для противника выборе одной из чистых стратегий на каждом ходе. Рассматривая любые смеси чистых стратегий в качестве новых стратегий, можно расширить матрицу игры без седловой точки, дополнив ее выбранными смесями чистых стратегий. Так, введя еще три стратегии для каждого игрока — в среднем три раза из четырех сказать (задумать) 1; в среднем половину раз сказать (задумать) 1;

в среднем один раз из четырех сказать (задумать) 1 — получим расширенную матрицу игры в "Чет-нечет" (табл. 11).

Дополнительные элементы матрицы в подобных случаях вычисляют по формуле $a_{ij} = p_{y1}(p_{x1}a_{11} + p_{x2}a_{21}) + p_{y2}(p_{x1}a_{12} + p_{x2}a_{22})$, где p_{xi} и p_{yi} — относительные "доли" i -й стратегии, используемые соответственно первым и вторым игроками. Например, для элемента a_{33}

Таблица 11

Расширенная матрица игры "Чет-нечет"

		Игрок <i>B</i>					
Игрок <i>A</i>		y_1 (зад.1)	y_2 (зад.0)	$\frac{3y_1 + y_2}{4}$	$\frac{y_1 + y_2}{2}$	$\frac{y_1 + 3y_2}{4}$	$\min a_{ij}$
$x=1$ ("сказать 1")		1	-1	0,5	0	-0,5	-1
$x=0$ ("ека- зать 0")		-1	1	-0,5	0	0,5	-1
$\frac{3x_1 + x_2}{4}$		0,5	-0,5	0,25	0	-0,25	-0,5
$\frac{x_1 + x_2}{2}$		0	0	0	0	0	0
$\frac{x_1 + 3x_2}{4}$		-0,5	-0,5	-0,25	0	0,25	-0,5
$\max a_{ij}$		1	1	0,5	0	0,5	

матрицы $p_{x_1} = p_{y_1} = 0,75$; $p_{x_2} = p_{y_2} = 0,25$ и, следовательно, $a_{33} = 0,75 (0,75 \cdot 1 + 0,25(-1)) + 0,25(0,75(-1) + 0,25 \cdot 0,5) = 0,25$. Выбрав в расширенной матрице минимальные элементы строк и максимальные элементы столбцов, убеждаемся в появлении седловой точки при равной нулю цене игры, следовательно, игра "Чет-нечет" справедлива.

Интересно отметить, что если игрок придерживается оптимальной стратегии, соответствующей седловой точке, то теоретически его выигрыш в среднем равен цене игры и не зависит от выбора стратегий противником. Игрок *A*, заметив, что игрок *B* не придерживается оптимальной стратегии, может попытаться использовать это обстоятельство, но это связано с риском, так как и игрок *B* также поступит аналогично. В подобных играх большую роль играет психология их участников. Человеку (даже если он рассеян и не отличается аккуратностью) без помощи вспомогательных средств практически невозможно составить действительно случайную последовательность ходов в требуемой пропорции. Поэтому в достаточно длинной серии ходов человека можно обнаружить определенные закономерности (обычно связанные не со средним числом различных стратегий, а с порядком их использования) в поведении противника и использовать их для выигрыша.

Хорошим примером может служить рассказ Эдгара По "Украденное письмо", герой которого объяснил свои победы в игре "Чет-нечет" примерно следующим образом: "Если я играю с тугодумом, то знаю, что он думает, что я следующим ходом изменю четность — я ее сохраняю и выигрываю. Если же играю с хитрецом, то знаю, что он думает, что я, зная о его хитрости, не изменю четности — я ее изменяю и выигрываю".

Приписывать микрокалькулятору понимание человеческой психологии, по крайней мере, рискованно, но микрокалькулятор можно обучить предсказанию следующего

хода человека на основании его предыдущих ходов. В играх с двумя чистыми стратегиями, подобными игре "Чет-нечет", преобладающую стратегию противника можно оценивать статистическими методами. Для этой цели, в частности, пригодна автокорреляционная функция (см. § 2.4) выбора человеком последовательности одного из двух решений. Подобный метод использован в следующей программе.

Программа 1 игры "Чет-нечет"

sin	$x \geq 0$	F↑	C _x	↑	2	X	1	—	↑	F4	X
→	F3	P4	X	→	F2	P3	X	XY	P2	F5	+
P5	↑	F2	X	←	↑	F6	+	P6	↑	F3	X
←	↑	F7	+	P7	↑	F4	X	↑	→	+	↑
→	+	$x \geq 0$	F9	1	БП	P+	C _x	C/P	БП	F↑	

Перед началом игры память микрокалькулятора очищают, в регистр X заносят любое многозначное число и нажимают клавиши В/О и С/П, после чего на индикаторе высвечивается ответ 0 (предсказание следующего хода противника). После этого в регистр X вводят ход y_i (0 или 1) противника и нажимают только клавишу С/П, что приводит к вычислению хода x_{i+1} микрокалькулятора, предсказующего ход противника. В процессе игры в регистрах памяти накапливается информация о предыдущих ходах.

Практически следует заранее составить и записать на бумаге последовательность ходов противника, чтобы исключить коррекцию его поведения в зависимости от предсказаний микрокалькулятора. Микрокалькулятор с рассмотренной программой несложно обыграть (по сумме угаданных ходов) на первых порах, но по мере изучения им Вашей психологии победа над микрокалькулятором будет даваться все труднее, а при достаточно длинных последовательностях он будет уверенно Вас обыгрывать.

При вводе 1000 = РХ перед первым пуском программы и последовательности из 30 Ваших ходов 0011011100001101000011001001 предсказания микрокалькулятора имеют вид 100110101100001101000011001011 в первом раунде (предсказано правильно 16 ходов из 30) и 001100100001111001011110011011 (20 ходов из 30) во втором раунде – по мере накопления информации микрокалькулятор выигрывает с большим успехом.

В рассмотренной программе вычисляются три коэффициента корреляции с учетом трех предыдущих ходов противника. Следующая программа для участия микрокалькулятора в качестве угадывающего игрока в игре "Чет-нечет" вычисляет четыре коэффициента корреляции и при выборе своего хода учитывает четыре предыдущих хода противника.

Программа 2 игры "Чет-нечет"

↑	2	X	1	–	P8	←	←	C _x	P7	–	ПП
8	F5	+	P5	ПП	P7	F4	+	P4	ПП	P7	F3
+	P3	ПП	P7	F2	+	P2	ПП	P7	F7	$x \geq 0$	6
1	ВП	FBП	C _x	C/П	БП	P0	↑	F6	X	↑	F7
+	P7	F6	↑	F8	X	XУ	←	P6	B/O		

Пользование этой программой аналогично предыдущей, за исключением того, что перед первым пуском в регистр X вводят 0 или 1. В среднем эта программа играет более успешно, но обучение микрокалькулятора длится несколько дольше – при повторных вводах рассмотренной выше последовательности из 30 ходов противника микрокалькулятор первый раз угадывает 16 ходов из 30, второй раз – 17, третий – 19 и четвертый – 20 ходов.

Читатель должен преисполниться еще большего почтения к микрокалькулятору, угадывающему ход противника в игре "Чет-нечет", если узнает, что использованная выше пробная последовательность его ходов составлена с помощью программы, генерирующей случайный выбор

чисел 0 и 1 с равной вероятностью. Конечно, стратегия с равной вероятностью решений, как мы уже знаем, оптимальна и при достаточно большом числе ходов микрокалькулятор не должен выигрывать. "Секрет" заключается в том, что в этой сравнительно короткой последовательности из 30 ходов доли чистых стратегий несколько отличаются от 0,5 и, кроме того, при повторении одной и той же последовательности она перестает быть случайной.

Рассмотрим еще одну игру, сводимую к игре с матрицей 2×2 . Первый игрок A , записав 0 или 1, сообщает игроку B правду или ложь о своем выборе. Выигрыш игрока A (и проигрыш игрока B) при ответах игрока B "верю" или "не верю" оценивают элементами следующей таблицы:

	B_1 ("верю")	B_2 ("не верю")
A_1 ("записал 1, сказал 1")	0	2
A_2 ("записал 0, сказал 1")	1	-2
A_3 ("записал 1, сказал 0")	-1	-2
A_4 ("записал 0, сказал 0")	-1	2

Хотя формально игрок A располагает четырьмя стратегиями, из них только A_1 и A_2 являются доминирующими. Стратегия A_3 проигрышна для игрока A при любом ответе игрока B , а стратегия A_4 слишком рискованна, так как при утверждении ("записал 0") для игрока B выгодна лишь стратегия B_1 , при которой игрок A проигрывает. Поэтому практически эта игра сводится к игре со следующей матрицей размером 2×2 :

	B_1 ("верю")	B_2 ("не верю")
A_1 ("записал 1, сказал 1")	0	2
A_2 ("записал 0, сказал 1")	1	-2

Эта матрица, как и матрица игры в "Чет-нечет" (в отличие от матрицы ранее описанной "игры" адмиралов) не содержит седловой точки — верхняя 1 и нижняя 0 цены игры не совпадают. В этом случае, очевидно, оптималь-

ной будет смесь стратегий, которую мы попытаемся оценить без громоздкого расширения матрицы игры.

При оптимальной смеси чистых стратегий средний выигрыш не должен зависеть от выбора стратегии противником. Поэтому доля p_1 чистой стратегии A_1 первого игрока в смеси может быть найдена из условия равенства среднего выигрыша как при стратегии "верю", так и при стратегии "не верю", или $a_{11}p_1 + a_{21}(1-p_1) = a_{21}p_1 + a_{22} \times (1-p_1)$, откуда $p_1 = (a_{22} - a_{21}) / (a_{11} + a_{22} - a_{12} - a_{21}) = -3 / (0-2-1-2) = 0,6$ и $p_2 = 1 - p_1 = 1 - 0,6 = 0,4$.

Аналогично, доля q_1 стратегии B_1 в оптимальной смеси чистых стратегий второго игрока определяется равенством $a_{11}q_1 + a_{12}(1-q_1) = a_{21}q_1 + a_{22}(1-q_1)$, откуда $q_1 = (a_{11} - a_{12}) / (a_{11} + a_{12} - a_{12} - a_{21}) = (-2-2) / (0-2-1-2) = 0,8$ и $q_2 = (1 - q_1) = 0,2$.

Полученные буквенные соотношения справедливы для любой игры с матрицей 2×2 (с двумя чистыми стратегиями). Цена Ц такой игры соответствует оптимальной смеси стратегий: $\text{Ц} = a_{11}p_1 + a_{12}(1-p_1) = a_{21}p_1 + a_{22}(1-p_1) = a_{11}q_1 + a_{12}(1-q_1) = a_{21}q_1 + a_{22}(1-q_1) = a_{11}p_1q_1 + a_{12}p_1(1-q_1) + a_{21}(1-p_1)q_1 + a_{22}(1-p_1)(1-q_1)$. Цена рассматриваемой игры Ц = $0 \cdot 0,6 + 1 \cdot (1 - 0,6) = 0,4$, и, следовательно, при принятой оценке сочетаний стратегий эта игра несправедлива — при большом числе ходов и оптимальных стратегиях обоих игроков первый из них выигрывает в среднем 0,4 балла на каждом ходе.

После определения оптимальной смеси стратегий игрок должен обеспечить и случайность последовательности своих ходов — в противном случае его партнер может воспользоваться замеченной закономерностью ходов в своих интересах. Поэтому для участия в рассмотренной игре программируемого микрокалькулятора программа игры должна содержать подпрограмму или фрагмент, генерирующий в случайном порядке чисел 0 и 1 с требуемой вероятностью. Подобная подпрограмма необходима и при

участии микрокалькулятора в качестве запаздывающего игрока в игре "Чет-нечет".

Прежде чем перейти к программам, генерирующими случайные числа, предлагаем читателю принять участие в отборочных соревнованиях для чемпионата мира среди микрокалькуляторов по игре "Чет-нечет", выполнив следующие задания:

1. Сформируйте для каждой из приведенных программ достаточно длинные последовательности, в которых микрокалькулятор ни разу не выиграл. Для этого, дождавшись очередного ответа микрокалькулятора 0 или 1, "обманите" микрокалькулятор, утверждая, что микрокалькулятор не угадал, и вводя в регистр X в качестве задуманного число 1 или 0, дополнительное к ответу микрокалькулятора.

2. Дайте каждую из этих дополняющих последовательностей на "разгадку" другой программе и оцените число взаимных выигрышей программ друг у друга.

3. Если Вы временно располагаете двумя микрокалькуляторами "Электроника Б3-21", то устройте их состязание в игре "Чет-нечет" с различными программами, взяв на себя роль посредника, обеспечивающего обмен данными между микрокалькуляторами.

4.5. Игры со случаем

Многие игры, кроме личных ходов, включают и случайные ходы, результат которых не зависит от участников, а определяется некоторым механизмом случайного выбора. Классическим примером таких игр, называемых азартными, является игра с подбрасыванием монеты одним игроком и угадыванием вторым положения (вверх аверсом — "орлом" или реверсом — "решкой") упавшей монеты. Весьма древней азартной игрой является и игра в кости, заключающаяся в поочередном бросании игроками одного или нескольких кубиков (некогда изготавливавшихся из кости), грани которых пронумерованы числами

от 1 до 6, с выигрышем участника, "выбросившего" наибольшее число. Современными вариантами игры в кости являются всевозможные детские игры, в которых очередной ход участников случаен и зависит от результатов бросания кубика с числами на гранях.

Многие сложные азартные игры, например бридж или преферанс, кроме случайных ходов (сдачи карт) содержат и личные. Но и в этих играх при одинаковом умении участников (заключающемся в одинаковом знании и использовании оптимальных стратегий) результат конкретной партии зависит от случая.

Случайные числа, соответствующие случайным положениям упавших монеты и кубика, можно получать на микрокалькуляторе с помощью специальных программ. При использовании программируемого микрокалькулятора эту задачу проще всего решить, связав случайность полученного результата со случайностью остановки непрерывно выполняемой программы. Например, для получения двух или нескольких различных чисел случайным образом достаточно составить программу автоматических вычислений без оператора остановки, заносящую требуемые числа поочередно в регистр X . При остановке такой программы нажатием клавиши С/П в случайный момент времени высвечиваемое на индикаторе число также будет случайным.

Выбрав, например, числа 0 и 1 в качестве аналогов двух равновероятных событий (с вероятностью 0,5 каждого из них), можно заполнить программную память последовательностями вида 1 С_x 1 С_x ... Тогда остановка программы в случайный момент времени приведет к высвечиванию числа 0 или 1 с равной вероятностью. Введя предварительно числа 0 и 1 в регистры X и Y , можно также заполнить программную память операторами XY — тогда при остановке программы в случайный момент времени высвечиваемое число 0 или 1 также будет случайным,

причем подобная программа хорошо моделирует операторами XY вращение бросаемой монеты.

Подобным образом можно имитировать бросание монеты, записав предварительно числа 0 и 1 в любые два регистра памяти и составив программу из операторов вызова содержимого этих регистров (например, F2 и F3). При этом следует учесть возможность нарушения равной вероятности появления генерируемых символов 0 и 1 при остановке программы через примерно одинаковые промежутки времени в связи с тем, что время вызова из памяти или выполнения другой простейшей операции над содержимым регистров памяти и операционного стека в микрокалькуляторе "Электроника Б3-21" составляет около 0,1 с. Эта "несправедливость" устраняется при выборе близкого к случайному чередованию операторов программы, генерирующих символы. Примером может служить следующая программа.

Программа моделирования бросания монеты

F2	F2	F3	F2	F3	F3	F2	F3	F3	F3	F2	F2
F2	F2	F3	F3	F2	F3	F3	F3	F3	F2	F2	
F2	F3	F2	F3	F3	F3	F3	F2	F2	F2	F2	
F3	F2	F3	F2	F3	F3	F3	F2	F2	F3	F2	F2
F3	F2	F2	F3	F3	F2	F3	F2	F2	F3	F2	F3

Если необходимо получить случайные состояния с вероятностью, отличающейся от 0,5, то следует изменить в такой программе доли операторов, вызывающих из памяти соответствующие символы. Так, для появления числа, хранящегося в регистре N , с вероятностью p_N программа должна содержать количество операторов FN, равное округленному до ближайшего целого числа значению $60p_N$.

Бросание кубика с числами от единицы до шести на его гранях можно имитировать аналогичным способом, записав эти числа в регистры 2–7 соответственно и воспользовавшись следующей программой без оператора С/П с

близким к равномерному распределению оператора вызова содержимого указанных регистров.

Программа игры в кости

F2	F7	F3	F6	F4	F5	F3	F6	F4	F5	F2	F7
F6	F3	F7	F2	F5	F4	F4	F5	F6	F3	F7	F2
F4	F2	F5	F7	F3	F6	F3	F6	F4	F5	F2	F7
F4	F6	F3	F7	F2	F5	F6	F3	F7	F2	F5	F6
F7	F2	F5	F4	F6	F3	F4	F5	F2	F7	F3	F4

Многие, если не большинство, азартные игры несправедливы, принося в среднем выигрыш только одной стороне. Эксплуатация надежд на возможность легко разбогатеть при "везении" в азартной игре привела к созданию в капиталистических странах целой отрасли бизнеса, связанной с автоматами для азартных игр. Игрок бросает в такой автомат монету и дергает за расположенную сбоку рукоятку, из-за которой подобные автоматы и получили меткое название "одноруких бандитов": Если игроку повезет, то автомат "отвалит" ему выигрыш, если нет — монета пропала для игрока. Бесстрастный автомат, подчиняясь железным законам теории вероятности, глотает монету за монетой, изредка отдавая часть их счастливчикам, но безотказно набивая карманы владельцу автомата. В бесперспективности длительной игры с "одноруким бандитом" легко убедиться, промоделировав эту азартную игру с помощью программируемого микрокалькулятора. Пусть за пять ходов (бросаний монет) автомат в среднем выигрывает одну монету, отдавая остальные в виде выигравшей в 2, 3 и 4 монеты. Определим число проигравших, приходящихся на каждый из таких выигравших: на выигрыш двух монет должно приходиться $3/2$ проигравших (в двух из пяти ходов автомат отдает по 2 монеты); на выигрыш трех монет должно приходиться $11/4$ проигравших (за 15 ходов автомат отдает четыре

выигрыша по 3 монеты); на выигрыш четырех монет должно приходиться четыре проигрыша в пяти ходах.

Запишем выигрыш в 2, 3 и 4 монеты соответственно в регистры 2, 3 и 4, а проигрыш (0) – в регистр. Распределив надлежащим образом в программной памяти два оператора F4, четыре оператора F3, 14 операторов F2 и 40 операторов F8, составим программу игры с "одноруким бандитом", роль которого безропотно придется выполнять микрокалькулятору.

Программа моделирования "безрукого бандита"

F8	F8	F3	F8	F8	F8	F2	F2	F8	F8	F8	F8	F8
F4	F2	F8	F8	F8	F8	F2	F8	F3	F2	F8	F8	
F8	F8	F2	F8	F8	F4	F8	F2	F8	F8	F3	F8	
F8	F8	F8	F2	F2	F8	F8	F3	F8	F8	F2	F8	
F2	F8	F8	F8	F8	F2	F8	F8	F2	F2	F8	F8	

Пустив эту программу и через некоторое время остановив ее нажатием клавиши С/П, Вы увидите на индикаторе число выигранных монет или, скорее, число 0, указывающее на проигрыш одной монеты. Возможно, что несколько раз Вам повезет и Вы обыграете "безрукого бандита". Однако чем дольше Вы будете играть, тем большим будет проигрыш и, сыграв достаточно большое число партий, Вы сможете убедиться, что участие в азартных играх такого рода в надежде на "везение" явно не окупается.

Основной недостаток приведенных выше программ-генераторов случайного появления чисел заключается в невозможности их сопряжения с другими программами. Однако последовательность случайных (точнее – квазислучайных или "как бы случайных") чисел можно получить и путем циклического повторения определенного вычислительного алгоритма. В частности, экспериментальные исследования свидетельствуют, что достаточно хорошие вероятностные характеристики имеет следующая упрощенная программа-генератор случайных чисел.

Генератор чисел, равномерно распределенных в интервале [0; 1]

π 1 – ↑ F4 ← – x^2 x^2 P5 1 ВП
7 XY + XY – ↑ F5 XY – P4 С/П БП
P0

Перед использованием этой программы в регистр 4 и во все регистры стека заносят любые различные числа из интервала [0; 1]. Для первых выпусков микрокалькулятора "Электроника Б3-21" эту программу следует изменить следующим образом:

π 2 – ↑ F4 ← + x^2 x^2 P5 1 ВП
7 XY – – ↑ F5 – ↑ F6 + P4 С/П
БП P0

В этом случае, кроме заполнения регистров стека памяти и регистра 4 ЗУПВ, следует занести в регистр 6 число $4/9 = 0,44444444$. После ввода этих программ в программную память каждое нажатие клавиши С/П (первый раз – В/О и С/П) приводит к высвечиванию на индикаторе очередного квазислучайного числа, вычисленного по формуле $x_i = (\pi - 2 + x_{i-7})^4 - E((\pi - 2 + x_{i-7})^4)$, где символом E обозначено выделение целой части выражения в скобках.

Во многих вероятностных задачах возникает необходимость в генерировании двух вариантов случайных событий с определенной вероятностью. Решение этой задачи обеспечивает следующая программа.

Программа-генератор событий 0 и 1 с задаваемой вероятностью

F2 sin 9 × ↑ F4 – $x \geq 0$ P2 1 БП F2
0 P8 F3 P2 – 9 + P3 F8 С/П БП P0

Перед использованием этой программы в регистры 2 и 3 заносят два различных произвольных числа, а в регистр 4 – число $k = 9 \cos(\pi p(1))$, где $p(1)$ – требуемая вероятность

генерирования символа 1. После нажатия клавиши С/П (первый раз – В/О и С/П) программа высвечивает символ 1 при выполнении условия

$$9 \sin \varphi_{i+1} - k \geq 0 \text{ (где } \varphi_{i+1} = 9 \sin \varphi_i - k - \varphi_i + 9)$$

или символ 0, когда это условие не выполнено.

При игре в "Чет-нечет" оптимальная стратегия соответствует задумыванию и угадыванию символа 1 с вероятностью 0,5. В этом случае $k = 0$ и приведенную программу-генератор можно упростить следующим образом.

Программа задумывания чисел в игре "Чет-нечет"

F2	sin	9	X	$x \geq 0$	1	1	БП	FXY	0	P8	F3
P2	-	9	+	P3	F8	С/П	БП	P0			

При игре в "Чет-нечет" с этой программой пользователь вводит в регистры 2 и 3 различные числа и, записывая свое предсказание хода микрокалькулятора (0 или 1), нажимает клавишу С/П (первый раз – В/О и С/П).

Читатель, внимательно ознакомившись с приведенным описанием игры "Чет-нечет", сможет выбрать оптимальную стратегию для игры "на равных" с микрокалькулятором, но рассчитывать на успех в достаточно длинных партиях ему не приходится.

В заключение составим программу игры "Веришь – не веришь" со следующими правилами. Первый игрок *A* вынимает из перетасованной колоды карточек (половина которых обозначена нулями и половина – единицами) одну и, не показывая ее противнику, утверждает (личный ход игрока *A*) "я вытянул единицу (или нуль)". Если игрок *B* отвечает "верю", то он выигрывает один балл при утверждении "я вытянул нуль" и проигрывает один балл при утверждении "я вытянул единицу". Требование проверки (ход "не верю" игрока *B*) оценивают как выигрыш двух баллов первым игроком, если его утверждение истинно, и проигрыш им двух баллов при ложном утверждении.

Эти условия игры отображаются следующей таблицей:

	B_1 ("верю")	B_2 ("не верю")
A_1 ("вытянув 1, сказать 1")	1	2
A_2 ("вытянув 1, сказать 0")	-1	-2
A_3 ("вытянув 0, сказать 1")	1	-2
A_4 ("вытянув 0, сказать 0")	-1	-1

В этой таблице выражение может вызвать элемент $a_{42} = -1$, который по условию (при стратегиях A_4 и B_2) должен быть равным 2. Причина коррекции заключается в бесмыслиности стратегии A для игрока A , так как она всегда проигрышна. Поэтому утверждение игрока A "я вытянул 0" будет истинным и игроку B выгодно использовать лишь стратегию "верю" с минимальным проигрышем.

Предполагая, что микрокалькулятор участвует в этой игре в качестве игрока A , определим его оптимальную стратегию при достаточно большом количестве партий. Вытянув единицу, игроку A достаточно стратегии A_1 , так как в этом случае он выигрывает при любой стратегии игрока B . Однако если он вытащит нуль, то ему нужно иметь две стратегии A_3 и A_4 . Поэтому при многоходовой игре игрок A практически располагает только двумя стратегиями: A_1 — "сказать 1", независимо от того, что он вытянул (по существу, это смесь стратегий A_1 и A_3), и A_2 — "сказать правду" (или смесь стратегий A_1 и A_4).

С учетом вероятностей исходов случайного хода (вытягивания нуля p_0 или единицы p_1) элементы матрицы рассматриваемой игры в этом случае определяются в соответствии с ранее приведенными формулами как $a'_{11} = a_{11}p_1 + a_{31}p_0$; $a'_{12} = a_{12}p_1 + a_{32}p_0$; $a'_{21} = a_{12}p_1 + a_{42}p_0$. Приняв $p_0 = p_1 = 0,5$, получим матрицу игры "Веришь — не веришь" со следующими элементами:

	B_1 ("верю")	B_2 ("не верю")
A_1 ("сказать: вытянул 1")	1	0
A_2 ("сказать правду")	0	0.5

Эта матрица не имеет седловой точки и, следовательно, оптимальной для обоих игроков будет смешанная стратегия. "Пропорцию" при использовании игроком стратегий A_1 и A_2 несложно определить по приведенным ранее формулам: $p_1 = 0,5(1 + 0,5 - 0 - 0) = 1/3$; $p_2 = 1 - p_1 = 2/3$. Цена игры $\Pi = 1/3$ и, следовательно, игра несправедлива и при выборе оптимальной стратегии выгодна микрокалькулятору.

Используя модифицированный генератор случайных чисел 0 и 1, составим следующую программу для участия микрокалькулятора в игре "Веришь – не веришь" в качестве игрока, вытягивающего карточки.

Программа игры "Веришь – не веришь"

F2	sin	9	$X \geq 0$	1.	1	БП	FXY	0	P7	F3	
P2	–	9	+	P3	cos	2	I/x	–	$x \geq 0$	P,	1
БП	4	F7	C/P	$x \neq 0$	7	F7	$x \neq 0$	FBП	F5	1	+
P5	C/P	БП	P0	F5	1	–	БП	P6	F7	C/P	$x \neq 0$
F–	F5	2	БП	/–	F5	2	БП	P7			

В этой программе генерируемый случайный символ заносится в регистр 7, а следующий генерируемый символ определяет стратегию микрокалькулятора – если "выпал" 0, то микрокалькулятор выбирает стратегию "сказать: вытянул 1", если же "выпала" 1 – стратегию "сказать правду".

Перед пуском программы следует занести в регистры 2 и 3 различные числа и очистить регистр 5. Пуск программы нажатием клавиши С/П (первый раз – В/О и С/П) приводит к высвечиванию 0 или 1. Если противник выбирает стратегию "верю", то он вводит или сохраняет в регистре X единицу и нажимает клавишу С/П, что приводит к высвечиванию суммарного выигрыша микрокалькулятора. Если противник выбирает стратегию "не верю", то он должен ввести (или сохранить) нуль в регистр X и нажать клавишу С/П, после чего на индикаторе высветится "вы-

тянутое" микрокалькулятором число 0 или 1, а после повторного нажатия клавиши С/П – суммарный выигрыш микрокалькулятора, полученного им в соответствии с правилами игры.

Для участия микрокалькулятора в этой игре в качестве игрока *B* достаточно использовать программу-генератор чисел 0 и 1 с требуемой вероятностью высвечивания символа 1.

Рассмотренные примеры выбора оптимальных стратегий в играх с личными и случайными ходами читатель может использовать для самостоятельного участия программируемого микрокалькулятора любого типа (или, при выполнении программ в обычном режиме, и непрограммируемых микрокалькуляторов) во многих других играх.

СПИСОК ЛИТЕРАТУРЫ

1. Гарднер М. Математические головоломки и развлечения. – М.: Мир, 1971. – 510 с.
2. Гарднер М. Математические новеллы. – М.: Мир, 1974. – 454 с.
3. Демидович Б. П., Марон А. И. Основы вычислительной математики. – М.: Наука, 1966. – 664 с.
4. Доморяд А. П. Математические игры и развлечения. – М.: Физматгиз, 1961. – 268 с.
5. Иванов В. В. Чет и нечет. Асимметрия мозга и знаковых систем. – М.: Сов. радио, 1978. – 184 с.
6. Кнут Д. Искусство программирования. Том 1. – М.: Мир, 1977. – 874 с.
7. Трохименко Я. К., Любич Ф. Д. Инженерные расчеты на микрокалькуляторах. – Киев: Техника, 1980. – 384 с.
8. Трохименко Я. К., Любич Ф. Д. Радиотехнические расчеты на микрокалькуляторах. – М.: Радио и связь, 1983.

ЯРОСЛАВ КАРПОВИЧ ТРОХИМЕНКО
ФЕЛИКС ДМИТРИЕВИЧ ЛЮБИЧ

"МИКРОКАЛЬКУЛЯТОР, ВАШ ХОД!"

Редактор В. И. К о т и к о в

Художественный редактор Н. С. Ш е и н

Обложка и заставки художника В. Н. З а б а й р о в а

Технический редактор Г. З. К у з н е ц о в а

Корректор З. Г. Г а л у ш к и н а

ИБ № 279

Подписано в печать 19.03.85 Т-07635 Формат 70x100/32

Бумага офс. № 1 Гарнитура "Пресс-роман" Печать офсетная

Усл. печ. л. 9,1 Усл. кр.-отт. 36,4 Уч.-изд. л. 9,41 Тираж 100 000 экз.

Изд. № 19677 Заказ № 953 Цена 45 к.

Издательство "Радио и связь". 101000, Москва, Почтамт, а/я 693

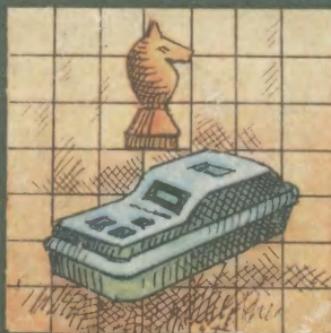
Типография издательства "Калининградская правда",
236000, г. Калининград обл., ул. Карла Маркса, 18

Список замечаний опечаток к книге Я. К. Трохименко, Ф. Л. Любича
"Микрокалькулятор, Ваш ход!"

На стр. 195 программа игры "Крестики-нолики" должна иметь следующий вид:

↑	9	↑	C/П	ПП	4	↑	π	X	cos	x < 0	F3
F2	ПП	4	1	-	БП	-	F7	ПП	4	F7	ПП
4	0	C/П	1	-	x=0	P/-/	8	P2	C/П	P7	F2
4	-	x ≠ 0	P7	x < 0	PC _x	8	+	↑	P8	F7	-
x=0	F-	F2	B/O	F8	7	7	C/П				

45 к.



В ЭТОЙ КНИГЕ РАССКАЗАНО
О ТОМ, КАК РАБОТАЮТ ЭЛЕКТРОН-
НЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ
НАЗЫВАЕМЫЕ МИКРОКАЛЬКУЛЯ-
ТОРАМИ, И О ТОМ, КАК "ОБУЧИТЬ"
МИКРОКАЛЬКУЛЯТОР РЕШАТЬ
ВСЕВОЗМОЖНЫЕ ЗАДАЧИ, И
О ТОМ, КАК САМЫЙ ПРОСТОЙ
НЕПРОГРАММИРУЕМЫЙ И
ОСОБЕННО ПРОГРАММИРУЕМЫЙ
МИКРОКАЛЬКУЛЯТОР МОЖЕТ
СТАТЬ НЕЗАМЕНИМЫМ ПОМОЩ-
НИКОМ В РЕШЕНИИ ПОВСЕДНЕВ-
НЫХ ПРАКТИЧЕСКИХ ЗАДАЧ -
ОТ НАИЛУЧШЕГО СПОСОБА
ПЕРЕСТАНОВКИ МЕБЕЛИ ДО
СОСТАВЛЕНИЯ ДИЕТИЧЕСКОГО
МЕНЮ ПО ВАШЕМУ ВКУСУ, И
О ТОМ, КАК МОЖНО ЗАНИМА-
ТЕЛЬНО И НЕ БЕЗ ПОЛЬЗЫ
ПРОВЕСТИ ДОСУГ В РАЗНООБ-
РАЗНЫХ ИГРАХ - ОТ УГАДЫВАНИЯ
ДНЯ РОЖДЕНИЯ ДО ТУРНИРОВ
НА ШАХМАТНОЙ ДОСКЕ С
МИКРОКАЛЬКУЛЯТОРОМ В РОЛИ
БЕСПРИСТАРСТНОГО СУДЬИ
ИЛИ РАВНОПРАВНОГО И ДАЖЕ
БОЛЕЕ УДАЧЛИВОГО ПАРТНЕРА.



«Радио и связь»