

Этот выпуск раздела наполнен отзвуками прошедшего лета.

Лето — время отдыха, время игр и забав. В этом выпуске раздела читатель найдет шуточный трактат о типах программистов, и разбор задачи, навеянной шахматной игрой. В большой программе на Паскале, приводимой двумя страницами далее, нет ни одной математической операции, а вместо Бейсика, о котором так много говорится в нашем разделе, на сей раз речь пойдет о языке Аналитик, все служебные слова которого взяты из русского языка и понятны без разъяснений.

АНАЛИТИК—ДРУГ ДЛЯ ДИАЛОГА

В начале 1968 года в маленькой комнате рядом с ВЦ нашего вуза появилась удивительная ЭВМ. Называлась она МИР, что означало «машина инженерных расчетов». Машина имела вид письменного стола с пишущей машинкой.

В комнату непрерывным потоком шли люди. Знатоки, выяснив, что быстродействие новой машины составляет лишь несколько сот операций в секунду, отходили с чувством превосходства: «Вот у нас ЭВМ дает десятки тысяч операций в секунду!». Студенты, аспиранты и инженеры просили дать им поработать на машине, а проводя какой-нибудь сложный расчет — например, вычислив сумму факториалов от 1! до 30! с 33 верными знаками, — приходили в восторг. Тем более, что программа такого вычисления записывалась просто и естественно: «РАЗРЯДНОСТЬ» 34.C = Σ (M = 1,30, π (K = 1, M, K)); «ВЫВОД» С»КОНЕЦ».

Особо привлекало то, что к МИРу придалось более 100 прикладных программ — таких, как вычисление специальных функций, решение систем линейных и нелинейных уравнений, действия с матрицами, интерполирование и аппроксимирование функций, интегрирование систем дифференциальных уравнений, решение дифференциальных уравнений в частных производных, статисти-

ческая обработка данных. Что касается вычисления интегралов по методу Симпсона, то это было просто включено в состав операций машины. Так, интеграл от функции $e^{-x} \cos x$ в пределах от 0 до $\pi/2$ вычисляется одним оператором $I = \int (X = 0, \pi/2, 8, e \uparrow (-X) \times \cos(X))$. Константы π и e заданы в машине с 20 разрядами.

Поскольку транслятор и операционная система МИРа «зашивались» в его постоянную память, машина не требовала настройки после включения. Клавиатура имела совмещенный русско-латинский алфавит, общепринятые математические символы Σ , π , \int , \geq , \leq , \neq , Π , e , ∞ , \uparrow и т. п. Все буквы находились на одном регистре, и это давало определенное удобство в работе.

Языку машины (он назывался Алмир-65) и операторской работе на ней никто не учил. В комнате висели два плаката, где разъяснялись операторы языка и порядок работы за пультом. Достаточно было несколько минут посмотреть на них, чтобы начать программировать и работать на машине.

Разработал эту удивительную ЭВМ в середине 60-х годов коллектив сотрудников Института кибернетики АН УССР под руководством академика В. М. Глушкова. Язык Алмир представляет собой развитие

ДОСТАТОЧНО ПЕРЕВЕСИТЬ «НОМЕРКИ»

На Паскале можно писать более эффективные программы, чем на Бейсике: они и выполняются быстрее, и памяти занимают меньше.

Возьмем для примера задачу о расстановке русских слов по алфавиту. Программа для ее решения, написанная на Бейсике, уже приводилась в «Науке и жизни» (№ 12, 1986 г., стр. 96). Посмотрим, как можно решить задачу средствами Паскаля.

Здесь перед нами открывается любопытная возможность: оказывается, чтобы упорядочить слова, вовсе не

обязательно передвигать их с одного места в памяти на другое, как это было в программе на Бейсике. Столь существенное упрощение предоставляет имеющееся в Паскале понятие ссылки. Используя его, мы, образно говоря, вешаем на каждое слово свой «номерок». Упорядочивание теперь сводится к перевешиванию «номерков» — а это и легче, и быстрее, чем перемещать слова.

Обратимся к помещенной на 68-й странице программе. Вот цикл, в котором слова, подлежащие упорядочиванию, вводятся в машину (строки 13—9 снизу). Он начинается со служебного слова REPEAT. А перед ним переменной I, счетчику ве-

денных слов, придается нулевое значение.

Что происходит с каждым выполнением цикла? Значение счетчика увеличивается на единицу, оператор WRITE высвечивает на экране подсказку, какое по счету слово вводится, оператор READLN считывает вводимое слово с клавиатуры.

Оператор READLN, как известно, присваивает вводимое значение той переменной, имя которой поставлено в скобках. Но тут в скобках стоит UKAZ[1] и далее — стрелка. То же самое имя стоит выше в скобках вслед за словом NEW.

Из этих деталей и складывается тот механизм, с помощью которого вводимые слова расставляются в

* О языке Паскаль рассказывалось в №№ 7, 9, 11 за 1987 г. под рубрикой «Школа начинающего программиста».

Алгола-60. В нем не объявляются типы величин, используемых в программе, нет ограничений на разрядность или на диапазон чисел. С неопределенными величинами производятся алгебраические действия и выводится результат в буквенном виде.

Вскоре к нам поступили машины МИР-1, которые имели ленточный перфоратор для запоминания программ и фотосчитыватель для их повторного ввода в ЭВМ. В 1969 году госкомиссия приняла новую машину МИР-2. Здесь появилось новое техническое достижение — дисплей для работы с программой; в качестве дисплея применялась электронно-лучевая трубка. Председатель комиссии академик А. А. Дородницын заявил («Известия» 7.12.69 г.), что МИР в несколько раз «умнее» больших ЭВМ, широко используемых в настоящее время.

Язык машины МИР-2 — Аналитик — явился дальнейшим развитием Алмира. Он приспособлен для работы в режиме диалога. Кроме обычных вычислений, он позволяет производить аналитические преобразования формул, интегрирование, дифференцирование, решать уравнения в буквенном виде.

На МИР-2 я начал работать в 1971 году и открыл для себя удивительные возможности языка Аналитик.

Мне надо было решить систему линейных алгебраических уравнений, но матрица системы оказалась плохо обусловленной и решение получалось абсурдным. Тогда я задал точность решения и построил итерационный процесс с автоматическим изменением разрядности вычислений. Ответ был получен точный, когда ЭВМ выбрала для вычислений 26 десятичных разрядов. Заметим, что на других ЭВМ такую задачу решить практически невозможно.

В 1978 году ко мне обратился мой друг, профессор, по специальности радиоэлектроник, с просьбой помочь его аспиранту составить на Фортране программу двумерного быстрого преобразования Фурье. Срок

аспирантуры кончался, и задачу надо было решить срочно. За два вечера я составил программу на Аналитике, в третий день отладил ее на МИР-2 и переписал на Фортране, а за четвертый и пятый день аспирант с моей помощью отладил программу на машине М-222. Если эту же задачу мне пришлось бы делать сразу на Фортране, то на нее ушло бы не менее месяца, так как надо было одновременно отлаживать алгоритм и программу.

Один мой знакомый полгода пытался вычислить на ЕС-1022 тройной интеграл. Им была перебрана масса вариантов решения, но ЭВМ выдавала только одно: «Исчезновение порядка». Я ему предложил решить эту задачу на МИР-2, но он скептически отклонил мое предложение: «Если большая ЭВМ не может решить задачу, то что можно ждать от малой?» Однако когда он окончательно разуверился в ЕС-1022, то пришел ко мне, хотя и имел твердое убеждение, что и на МИР-2 задачу не решить. За несколько минут мы набрали программу с оператором $P = \int (X = A, B, 8, \int (Y = B, 7, 8, \int (T = D, E, 8, F(X, Y, T)))$ и задали исходные данные. Примерно через полчаса машина отпечатала результат. Он оказался порядка 10^{-1215} . Естественно, во всех других ЭВМ, кроме МИР, такие числа изображаются нулем.

Сейчас в нашей стране возродился интерес к языкам машин семейства МИР. Как сказал ученый секретарь Совета МНТК «Персональные ЭВМ» В. Захаров («Правда», 21.01.88), если языки Алмир-Аналитик появятся в наших персональных компьютерах, то большинство проблем компьютерной грамотности у нас будет снято, поскольку многие трудности программирования в этих языках перенесены в транслятор, а не возлагаются на пользователя.

Л. ОСИПОВ
(г. Москва).

памяти машины вместе со своими «номерками». Понять его помогает хорошая особенность Паскаля: все переменные, участвующие в написанной на нем программе, предварительно характеризуются в разделе описаний. Обратимся туда.

УКАЗ: ARRAY [О.. ВСЕГОСЛОВ] OF ССЫЛКИ. И раньше: ССЫЛКИ = ↑ СЛОВА. Становится ясно: под обозначением УКАЗ[1] следует понимать «номерок» 1-го слова, под УКАЗ[1]↑ — само это слово.

Вот как работает механизм расстановки слов. Стандартная функция Паскаля NEW резервирует участок памяти для вводимой информации, обозначенной именем, которое стоит в скобках. Прочитав это имя, компьютер справляется в

разделе описаний VAR, что за ним стоит массив величин типа ССЫЛКИ. Компьютер отыскивает это слово в перечне типов, понимает, что стрелочка помещает ту переменную, на которую делаются ссылки, и ищет в том же перечне ее имя. СЛОВА = PACKED ARRAY [1.. БУКВСЛОВА] OF CHAR. В переводе на русский язык цепочка фигурирующих здесь служебных слов означает «упакованный массив символов».

Что значит упакованный? Дело в том, что в большинстве современных персональных компьютеров и мини-ЭВМ каждая ячейка памяти способна вместить 2 байта информации, а для записи одного символа достаточно одного байта. Отводить на каждый символ по

ячейке было бы расточительно, поэтому массивы символов (к которым относятся и слова) упаковывают плотнее. Об этом и говорит слово PACKED. (Надо заметить, на упаковку и распаковку требуется определенное время, что несколько замедляет выполнение программ.)

Каждое слово, как указано в квадратных скобках в описании типа СЛОВА, насчитывает такое число символов, какое задано константой ВСЕГОБУКВ, а оно, как указано при описании констант, равно 20. Стало быть, на хранение каждого слова будет отведено 10 байт. И когда очередное 1-ое слово будет считано с клавиатуры оператором READLN, именно такое место и займет в памяти перемен-

ВОСЕМЬ ФЕРЗЕЙ

«Владельцам микрокалькуляторов предлагаем составить программу, позволяющую расставить на шахматной доске 8 ферзей так, чтобы ни один не бил другого. Программа должна давать все варианты исковой расстановки» — такое предложение было помещено в одном из номеров нашего журнала за прошлый год.

Благодарим всех читателей, откликнувшихся на приглашение к конкурсу. Всего было прислано 102 письма.

Некоторые вообще не верили в саму возможность такой расстановки. Другие считают задачу непосильной для «БЗ-34». Третьи полагают, что «гридцать четверка» если и найдет все расстановки, то за несколько месяцев непрерывной работы. Однако самым настойчивым удалось преодолеть все трудности и полностью решить эту сложную задачу.

Алгоритм последовательного перебора вариантов (самый популярный из встречающихся в письмах) с точностью до несущественных различий выглядит следующим образом (обозначения вертикальных полей шахматной доски *abcdefgh* заменены цифрами 1—8):

ШАГ 1: поставить ферзей на горизонталь 1;

перейти к вертикали 2;

ШАГ 2: продвинуть ферзя на 1 клетку вверх;

ШАГ 3: ферзь вышел за пределы доски?

если да, то поставить его на первую клетку, вернуться к предыдущей вертикали, если при этом выходим за пределы доски, то КОНЕЦ;

иначе перейти на ШАГ 2;

ШАГ 4: данный ферзь бьется предшествующими?

если да, то перейти на ШАГ 2;

ШАГ 5: перейти к следующей вертикали; если вышли за пределы доски, то выдать полученную расстановку, перейти к вертикали 1;

перейти на ШАГ 2;»

Этот алгоритм реализовали наши постоянные читатели К. А. Лайва (г. Лимбажи Латвийской ССР), М. Л. Вулис (г. Москва), а также С. В. Вяткин (пос. Покотиловка Харьковской обл.), Дао Нгок Чунг (г. Хошимин, СРВ), А. Кадыров (г. Ленинград) и другие. Оригинальную программу, работающую по этому алгоритму, прислал Д. Л. Мамаев (п/о Кино Брянской обл.). Для задания положений фигур на горизонтальных он использовал ряд чисел 1, 10..., 10000000, что позволило передвигать ферзей операциями умножения и деления и сделало очень удобным математическую запись алгоритма.

Однако, несмотря на все ухищрения, среди этих программ не было ни одной, тратившей менее 1,5 часа (в среднем) на поиск одной расстановки. Где же выход?

Многие читатели обратили внимание на то, что из 92 расстановок, удовлетворяющих условию, только 12 являются независимыми, а остальные 80 получаются путем

```

PROGRAM SORT;
CONST БУКВСЛОВА=20;ВСЕГОСЛОВ=100;
TYPE СЛОВА=PACKED ARRAY[1..БУКВСЛОВА] OF CHAR;ССЫЛКИ=^СЛОВА;
VAR УКАЗ:ARRAY[0..ВСЕГОСЛОВ] OF СССЫЛКИ;
I,J,K,L:0..ВСЕГОСЛОВ;КОД:PACKED ARRAY[1..31] OF CHAR;
FUNCTION МЕНА(P,Q:СЛОВА):BOOLEAN;BEGIN
  J:=1;WHILE (P[J]=Q[J]) AND (J<БУКВСЛОВА) DO J:=J+1;
  IF P[J]>='0' THEN P[J]:=КОД[ORD(P[J])-95];
  IF Q[J]>='0' THEN Q[J]:=КОД[ORD(Q[J])-95];
  МЕНА:=P[J]>Q[J];END;
BEGIN(*НАЧАЛО ОСНОВНОЙ ПРОГРАММЫ*)
КОД:='ШЩАЖДЕТЦУХИЙКЛМНОЧПЯРСФБЗШГЬЭЯВ';I:=0;
REPEAT
  I:=I+1;NEW(УКАЗ[I]);WRITE(I,'-Е СЛОВО ');READLN(УКАЗ[I]^);
UNTIL EOF OR (I=ВСЕГОСЛОВ);
FOR K:=1 TO I-1 DO
  FOR L:=K+1 TO I DO
    IF МЕНА(УКАЗ[K]^,УКАЗ[L]^) THEN
      BEGIN УКАЗ[0]:=УКАЗ[K];УКАЗ[K]:=
        УКАЗ[L];УКАЗ[L]:=УКАЗ[0] END;
FOR K:=1 TO I DO WRITELN(УКАЗ[K]^) END.

```

ная $УКАЗ[I] \uparrow$. А переменная $УКАЗ[I]$ станет его «номерком». Ее значение — адрес первой из ячеек, отведенных под переменную $УКАЗ[I] \uparrow$.

Слово за слово... Так будет продолжаться до тех пор, когда I достигнет значения ВСЕГОСЛОВ, то есть

равенство $I=ВСЕГОСЛОВ$ станет истинным. Впрочем, это лишь одна из двух возможностей окончания ввода, как вытекает из строения заключительного оператора цикла: UNTIL EOF OR ($I=ВСЕГОСЛОВ$). Буквосочетание EOF расшифровывается как end of file, ко-

нец файла. Обозначаемая так, переменная не описывается в программе, компьютер всегда ее «имеет в виду» и присваивает ей в самом начале работы любой программы значение FALSE (ложь). Но если нажать одновременно клавишу <CV> (она находится

преобразования независимых решений поворотами шахматного поля на 90, 180 и 270 градусов, зеркальными отражениями относительно осей симметрии поля.

Вот где скрыта возможность повышения быстродействия: нужно найти 12 базовых вариантов, а из них уже получить все оставшиеся.

Помещенную рядом таблицу базовых вариантов прислал Н. Г. Соловьев (г. Тбилиси). Он провел большую и интересную работу по систематизации расстановок и нашел ряд свойств основных решений. Например, из варианта 1 можно получить два следующих циклическими сдвигами всех ферзей на одну клетку вниз; вариант 4 получается из варианта 1 аналогичным сдвигом на одну клетку вправо. Связь между вариантами 4, 5, 6 такая же, как и между тремя первыми, а вот общего принципа взаимосвязи всех вариантов найти так и не удалось. В приводимой ниже программе Н. Г. Соловьева для «МК-61» они получаются из исходного последовательными переходами с минимальным числом шагов. Аналогичную программу прислал М. В. Шуклин (Пермь).

00.8 01.8 02.X—Пе 03.3 04.X—Pd 05.5
06.X—Пб 07.2 08.X—Пс 09.8 10.X—Pd
11.КППе 12.2 13.X—П7 14.8 15.X—Пс 16.4
17.X—Pd 18.КППе 19.4 20.X—П6 21.5
22.X—П9 23.1 24.X—Па 25.8 26.X—Пб 27.6
28.X—Пс 29.3 30.X—Pd 31.КППе 32.8
33.X—П7 34.1 35.X—П8 36.7 37.X—Па 38.2
39.X—Пб 40.КППе 41.3 42.X—П9 43.6
44.X—Па 45.7 46.X—Пс 47.5 48.X—Pd

в левой стороне клавиатуры) и Z, переменная EOF принимает значение TRUE (истина). Логическое выражение из двух условий, соединенных союзом OR (или), истинно, когда истинно хотя бы одно из них. А оператор UNTIL прекращает выполнение цикла, когда записанное вслед за ним логическое выражение принимает значение TRUE.

Отныне значение переменной I равно числу введенных слов.

Цикл ввода закончен и начинается двойной (это видно по дважды написанному в его начале слову FOR) цикл упорядочивания введенных слов. Они последовательно берутся от первого до предпоследнего (FOR K: = 1 TO I—1) и каждое сравнивается со всеми, стоящими за ним (FOR L: = K+1 TO I). Функция МЕНА (о ней чуть позже) проверяет, подчиняется ли алфавитному порядку пара слов УКАЗ[K]↑ и УКАЗ[L]↑, и если не подчиняется, их «номерки» пе-

	1	2	3	4	5	6	7	8	9	10	11	12
(a) 1	4	3	2	6	5	4	4	4	4	5	1	6
(b) 2	1	8	7	4	3	2	2	2	8	1	7	4
(c) 3	5	4	3	1	8	7	7	7	1	4	4	7
(d) 4	8	7	6	5	4	3	3	5	5	6	6	1
(e) 5	2	1	8	8	7	6	6	1	7	8	8	8
(f) 6	7	6	5	2	1	8	8	8	2	2	2	2
(g) 7	3	2	1	7	6	5	1	6	6	7	5	5
(h) 8	6	5	4	3	2	1	5	3	3	3	3	3

Каждый столбец таблицы представляет вариант расстановки ферзей. Каждая цифра в столбце — это номер поля в вертикальном ряду клеток шахматной доски, задаваемая номером строки таблицы и соответствующим буквенным обозначением в скобках.

49.КППе 50.8 51.X—Пб 52.4 53.X—П7
54.КППе 55.2 56.X—П1 57.8 58.X—П0 59.5
60.X—П4 61.X—П5 62.КП—X4 63.1 64.—
55.FX=0 66.68 67.8 68.KX—П5 69.C/П
70.FL0 71.62 72.FL1 73.57 74.3 75.X—П7
76.2 77.X—П9 78.8 79.X—Па 80.5 81.X—Пб
82.1 83.X—Пс 84.4 85.X—Pd 86.БП 87.54
88.5 89.X—П4 90.8 91.X—П0 92.КП—X4
93.C/П 94.FL0 95.92 96.B/O.

Инструкция: ввести числа 6,4,7,1,8,2,5,3 в регистры 6...9 и A...D соответственно, B/O, C/П; для получения очередного варианта нажать C/П. Первой расстановкой является вводимая комбинация.

Ю. Г. Курсон (Львов), В. Волков (Москва) и некоторые другие считают, что 12 ос-

НОМЕР СТОЛБЦА	1	2	3	4	5	6	7	8
ЧИСЛОВОЙ КОД	96	97	98	99	100	101	102	103
КЛАВИАТУРА ЭВМ	Ю	А	Б	В	Г	Д	Е	Ж
АЛФАВИТ	А	Б	В	Г	Д	Е	Ж	З
ПРОИЗВОДНЫЙ КОД	Щ	Ю	А	Ж	Д	Е	Т	Ц

Схема поясняет образование производного кода, при помощи которого по приведенной выше программе расставляются в алфавитном порядке слова, написанные кириллицей.

Расположим буквы русского алфавита в три ряда, буква под буквой: сначала в том порядке, в котором они нанесены на клавиши компьютера, ниже — в алфавитном порядке, и начнем заполнять самый нижний ряд. Возьмем в верхнем буквенном ряду произвольную букву (на схеме — Ю). Найдем в среднем ряду ее «соседку» по вертикали (А). Отыщем «соседку» в верхнем ряду и точно под нею в самом нижнем ряду поставим взятую вначале букву.

Обратимся к программе. Стандартная функция ORD (P[J]) определяет код J-ой буквы слова P, разность [ORD(P[J])—95] дает номер позиции, на которой располагаются в нашей таблице и выбранная буква, и соответствующая ей буква производного кода. Замена первой из этих букв на вторую осуществляется «присваиванием P[J]: = [ORD(P[J])—95].

Так буква А (см. рисунок) заменится на букву Ю. Но «производная» буква Ю находится на том же (первом) месте на клавиатуре компьютера, что и исходная буква А в русском алфавите. Поэтому располагая «производные» буквы в порядке возрастания их кодов, мы одновременно будем упорядочивать исходные буквы по алфавиту.

новых решений найти легко и без машины и поэтому не стоит делать программу для их поиска. Они сосредоточили свои усилия на разработке программ для получения оставшихся 80 расстановок.

Победители конкурса — К. Бенёнис (Каунас) и С. Н. Банников (Москва) успешно объединили в одной программе для «БЗ-34» два алгоритма: один — для поиска основных расстановок, другой — для получения из каждой нескольких новых. Программа К. Бенёниса для каждого варианта, найденного последовательным перебором, находит семь производных и требует, по подсчету автора, в среднем, 3,3 минуты на вариант. 00.— 01.Кх≥0С 02.ФВх 03.КПД 04.+ 05.КП↑ 06.1 07.2 08.П1 09.КИП1 10.ИП1 11.ИПД 12.— 13.Фх≠0 14.43 15.≠ 16.КИПД 17.— 18.Кх≠0С 19.Фх² 20.Фу 21.— 22.Фх=0 23.09 24.КИПД 25.КИПО 26.Фх=0 27.00 28.≠ 29.КП↑ 30.ИПД 31.П1 32.1 33.+ 34.ПД 35.ПО 36.КИП1 37.КПО 38.Фх=0 39.36 40.ИПД 41.ПО 42.КБПС 43.ИПД 44.1 45.— 46.ПД 47.ПО 48.КИП↑ 49.Фх=0 50.06 51.1 52.2 53.ПО 54.КИПЗ 55.8 56.П1 57.Сх 58.ПЗ 59.1 60.0 61.× 62.КИПО 63.+ 64.ФЛ1 65.59 66.+ 67.ФВх 68.С/П 69.КБПС 70.8 71.П1 72.≠ 73.1 74.0 75.: 76.ПД 77.КИПД 78.≠ 79.ИПД 80.ПО 81.— 82.1 83.0 84.× 85.3 86.+ 87.ПД 88.ИП1 89.КПД 90.ИПО 91.7 92.Ф10× 93.+ 94.ФЛ1 95.73 96.БП 97.51.

Инструкция: 24 ПС 10 ПО ПД Сх П2 ПЗ 4 ПВ 8 ПА 7 П9 6 П8 5 П7 3 П6 2 П5 1 П4 БП 06 С/П «АААААААА» ≡ «—ВВВВ ВВВВ» ШГ вправо /—/ С/П «СССССССС»

реставляются: $УКАЗ[0] := УКАЗ[K]; УКАЗ := УКАЗ[L]; УКАЗ[L] := УКАЗ[0]$. Резервной ячейкой при этом служит нулевой элемент массива «номерков» $УКАЗ[0]$, который при вводе слов не получил никакого значения и остался свободным. Наконец, упорядоченный массив выводится на дисплей: $FOR K := 1 TO I DO WRITELN (УКАЗ[K])$.

Посмотрим теперь, как выполняется функция МЕНА. В ее описании вслед за ее именем стоит слово BOOLEAN. Стало быть, ее значения принадлежат к логическому типу и могут быть либо TRUE, либо FALSE. При обращении к функции МЕНА ее формальные параметры P и Q заменяются фактическими — словами $УКАЗ[K]↑$ и $УКАЗ[L]↑$. Но каждое слово — это массив символов. В программе, по которой вычисляется значение функции, эти символы пронумерованы с помощью индекса J. Они перебираются ($DO J := J + 1$), начиная с первого (присваивание $J := 1$

перед словом WHILE). Первые буквы обоих слов могут совпасть, и пока это так ($P[J] = Q[J]$) и пока не исчерпаны все буквы ($J < БУКВСЛОВА$), перебор продолжается. Согласно программе, далее происходит проверка неравенств $P[J] > 'Ю'$ и $Q[J] > 'Ю'$. Когда в неравенстве участвуют символы, сравниваются их коды. У буквы Ю он имеет наименьшее значение (см. рисунок) среди всех букв, у пробела — еще меньшее. Значит, невыполнение неравенства $P[J] > 'Ю'$ означает, что просмотрены все буквы слова и стоящий за этим неравенством оператор выполнять не следует. Но если элемент $P[J]$ представляет собой не пробел, а букву, тогда он заменяется элементом массива КОД — выполняется присваивание $P[J] := КОД[ORD(P[J]) - 95]$. Аналогично обрабатывается элемент $Q[J]$.

Зачем это делается? Дело в том, что на клавиатуре компьютера русские буквы расположены не в алфавит-

ном порядке, и их коды возрастают также не в соответствии с алфавитом. Но если заменить русские буквы на производные так, как показано на рисунке, то код каждой производной буквы будет соответствовать месту исходной буквы в алфавите.

Вот после этого уже можно сравнивать измененные величины $P[J]$ и $Q[J]$. Если первая больше второй, то есть выполняется условие $P[J] > Q[J]$, функция МЕНА принимает значение TRUE — его она и вернет в программу в то место, откуда произошло обращение к ней. Напомним: она вычисляется при проверке условия IF МЕНА ($УКАЗ[K]↑, УКАЗ[L]↑$), выполнение которого ведет к перестановке слов.

Эта программа эффективнее программы на Бейсике: она требует меньших перемещений информации в памяти машины и расставляет слова в алфавитном порядке, а не по первым буквам.

Каждый результат, записанный справа налево, дает еще четыре симметричные расстановки — всего получается 8 на один расчет основного варианта. Все 92 варианта получаются за 14 проходов основной части программы.

Программа, присланная С. Н. Банниковым, в каждом цикле находит одно решение методом перебора и отображение его относительно горизонтальной оси симметрии. 00.3 01.П1 02.2 03.ПА 04.7 05.КПА 06.ИПА 07.ПО 08.КИПО 09.КИПА 10.КИП↑ 11.— 12.Кх≠09 13.Фх² 14.Фу 15.ИПА 16.ИПО 17.— 18.— 19.Кх≠09 20.ФЛО 21.09 22.ИПА 23.1 24.+ 25.ПА 26.8 27.— 28.Фх≥0 29.04 30.7 31.ПО 32.ИПВ 33.КИП↑ 34.— 35.ФЛО 36.33 37.П8 38.7 39.ПО 40.ИП8 41.КИП↑ 42.— 43.Фх² 44.Фу 45.8 46.ИПО 47.— 48.— 49.Фх≠0 50.74 51.ФЛО 52.40 53.8 54.ПО 55.1 56.КИП↑ 57.+ 58.С/П 59.ФЛО 60.55 61.КНОП 62.8 63.ПО 64.8 65.КИП↑ 66.— 67.С/П 68.ФЛО 69.64 70.ИП7 71.3 72.— 73.П7 74.7 75.ПА 76.КИПА 77.1 78.— 79.Фх<0 80.05 81.ИПА 82.1 83.— 84.ПА 85.Кх=09 86.КСх.

Инструкция: 28 ПВ 76 П9 В/О С/П. После остановки на индикаторе — первая цифра расстановки, С/П — вторая, и т. д. Результаты выдаются попарно, без разделения. Среднее время на одну расстановку — 9 мин. Если же записать результаты в обратном порядке, как это делает Бенёнис, то за один проход программа будет выдавать четыре расстановки, и это время умень-

ном порядке, и их коды возрастают также не в соответствии с алфавитом. Но если заменить русские буквы на производные так, как показано на рисунке, то код каждой производной буквы будет соответствовать месту исходной буквы в алфавите. Вот после этого уже можно сравнивать измененные величины $P[J]$ и $Q[J]$. Если первая больше второй, то есть выполняется условие $P[J] > Q[J]$, функция МЕНА принимает значение TRUE — его она и вернет в программу в то место, откуда произошло обращение к ней. Напомним: она вычисляется при проверке условия IF МЕНА ($УКАЗ[K]↑, УКАЗ[L]↑$), выполнение которого ведет к перестановке слов.

Эта программа эффективнее программы на Бейсике: она требует меньших перемещений информации в памяти машины и расставляет слова в алфавитном порядке, а не по первым буквам.

В. ИВАНОВ
(г. Москва).

шится вдвое. Но обратите внимание: программная память использована не полностью, резерв есть; если бы автор находил еще и другие производные первой расстановки, среднее время на один вариант можно было бы сократить еще больше.

Кроме рассмотренных алгоритмов, в нашей почте встретились еще два. Некоторые читатели использовали метод Монте-Карло, выбирая с помощью генератора случайных чисел координаты ферзей до получения требуемой расстановки. Наиболее интересная программа такого рода принадлежит М. Л. Булису (Москва). К сожалению, этот метод не гарантирует получение всех результатов за обозримое время и потому не удовлетворяет условиям конкурса.

По другому пути пошел С. В. Шауро (Минск). Его алгоритм напоминает последовательный перебор. Однако, поставив фигуру на доску, он сразу определяет поля, которые лежат под боем этого ферзя: на эти поля следующая фигура уже не ставит-

ся. Вопреки ожиданиям метод не дал его автору выигрыша во времени.

Обзор читательских писем по просьбе редакции провел И. БЫСТРОВ (г. Москва).

● МАЛЕНЬКИЕ ХИТРОСТИ

На клавиатуре «МК-54» обозначения клавиш обмена информацией между регистром X и адресусмысленными регистрами чересчур громоздки: $P \rightarrow X$, $X \rightarrow P$. Я наклеил на них вырезанные из ватмана бирочки, на которых написал соответственно И и В, что означает: Из памяти, В память. Бирка И у меня зеленая, В — красная. Это вызвано последствиями нажатия этих клавиш. В красная потому, что нажатие ее и клавиши соответствующего регистра приводит к безвозвратной потере его содержимого. А при вызове информации из какого-либо регистра его содержимое сохраняется.

**И. ПЯТИЛЕТОВ
(г. Киев).**

ТИПЫ ПРОГРАММИСТОВ

Хотя программистов все еще мало, среди них есть уже свои типы. Я попытался охарактеризовать наиболее яркие из них.

КЛАССИФИКАЦИЯ С ТОЧКИ ЗРЕНИЯ ПОДХОДА К РАБОТЕ

Программист-«штамповщик» — добросовестный ремесленник в лучшем смысле этого слова. Работает, руководствуясь плотницким правилом «Десять раз отмерь, один раз отрежь». Программирование для него — честное ремесло. Что ни программа — то те же самые тщательно сработанные двери.

Программист-«богемщик» — бывает всесторонним художником. Программирование для него — конек и средства самовыражения. Его программы полны красок и тонов. Что ни программа — то произведение искусства. Что ни программа — то своя «Неоконченная симфония».

Программист-«стрелок», иногда называемый «очумелым» — продукт современной цивилизации. К программированию относится как к бою, его оружие — терминал. Что ни программа — то игра в тотализатор. Что ни программа — прыжок в воду.

КЛАССИФИКАЦИЯ С ТОЧКИ ЗРЕНИЯ ПРОФЕССИИ

Программист-рутинер — является результатом обычной профессиональной отупелости. Его не радуют плоды его работы, он не переживает каждый проход своей программы через компьютер, знает, что беганьем вокруг стола событий не ускорит. Взгляд на новый компьютер обрадует его не более, чем взгляд на новый кухонный гарнитур. Считает, что программировать умеет, и обычно оно так и есть.

Программист-любитель — хотя и работает в своей специальности уже лет 20, все еще не забыл тех счастливых минут, когда был сам себе компилятором, и перфоратором, и оператором в те времена, когда первый в республике УРАЛ-1 выдавал свои исторические результаты. В нем все еще горит юношеское волнение, когда через машину проходит именно его программа, а за каждый результат он боится, как за собственного ребенка. Часто думает, что программировать не умеет, но на самом деле все обстоит не так плохо.

Программист-дилетант — в сущности счастливый человек: все время чем-то восхищен, удивлен. Компьютер работает! Компьютер стоит! Солнышко светит! Он постоянно в движении, постоянно в деле. Все в поле его зрения, ничто не ускользнет от его глаз, ничто чужое ему не чуждо. Он убежден, что умеет программировать, но, как всегда, снова ошибается.

КЛАССИФИКАЦИЯ С ТОЧКИ ЗРЕНИЯ ХАРАКТЕРА

Программист-флегматик считает своим девизом «лучшее — враг хорошего». Он не разволнуется, если за неделю до окончания проекта выяснится, что можно было бы начать сначала и это было бы лучше. Знает, что хорошо закончить важнее, чем хорошо начать.

Программист-сангвиник мечется от здоровой неудовлетворенности. Свою жизнь проживает в собственных программах, и пока они несовершенны, спокойного сна у него не будет. Отличается кипучей активностью и хроническим недосыпанием.

Программист-холерик ничем не доволен. Одержим стремлением к полному совершенству, прогрессивности и полноте всего, поэтому переделывает уже переделанное и не доделывает недоделанное. Сомнения в его проекте рассматривает как наглость, а на вопрос, когда будет готов проект, впадает в обморочное состояние.

Подсчитаны 27 различных комбинаций типов программистов. Некоторые из них выглядят явно абсурдными. Остановимся хотя бы на трех крайностях:

«Штамповщик»-рутинер-флегматик — обычно любим своими шефами. Работает добросовестно, надежно, непогрешимо. В случае необходимости может проявить себя и как способный бухгалтер.

«Богемщик»-любитель-сангвиник — программист, каких поискать. В нем, как говорится, есть «искра божия», около него целый штаб помощников. Если его не свалит инфаркт, всегда готов для направления в психиатрическую лечебницу.

«Стрелок»-дилетант-холерик — известен в своем кругу под прозвищем «действительный болван». Совершенно не важно, что он делает в программировании. Он все равно не знает, как это делать.

И. ДЕМНЕР (журнал «Veda a tehnika mladezi»).

ВМЕСТО ДВУХ ВЫЧИСЛЕНИЙ — ОДНО

В июньском выпуске раздела «Человек и компьютер» за этот год предлагалось присылать в редакцию программы, форма листинга которых каким-то образом отражала бы сущность заложенного в них алгоритма или приема программирования. Думаю, что программа на Бейсике удовлетворяет этому требованию.

Если у вас есть знакомый художник, попросите его разделить произвольный отрезок на две неравные части. Возможно, он сделает это в золотом соотношении, если его чувство меры воспитано на классических образах.

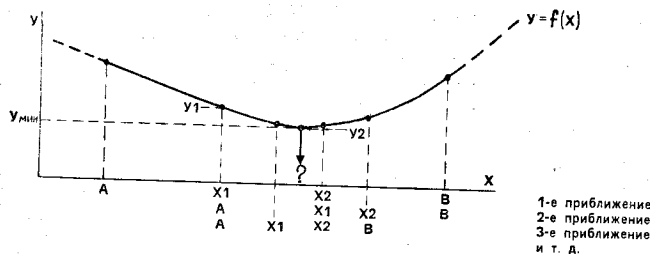
Такое чувство меры целесообразно внушить компьютеру, которому часто приходится отыскивать единственную точку минимума некоторой функции $F(X)$ на заданном отрезке $[A, B]$. Основа алгоритма — последовательное приближение к минимуму до достижения заданной точности ϵ . Сначала на исходном отрезке проставляются две точки

X_1 и X_2 , делящие его в золотом соотношении. Если $F(X_1) < F(X_2)$, то берем для дальнейшего деления отрезок $[A, X_2]$. Если же $F(X_1) > F(X_2)$, то берем $[X_1, B]$. В первом случае одной из двух делящих точек выбранного отрезка будет X_1 , во втором — X_2 . Это следует из свойств золотого сечения. Таким образом, из двух значений функции, необходимых на новом этапе расчета, вычисляется всего одно: другое было получено на предыдущем этапе.

Это компьютерное чувство меры особенно полезно в тех случаях, когда функция вычисляется по сложному алгоритму или определяется опытным путем.

Как принято в разделе «Человек и компьютер», я заключил программу на Бейсике в рамки структурной диаграммы. Размеры сторон диаграммы оказались точно в золотом соотношении. Потому я и посылаю их вам в ответ на ваше предложение.

В. ВОРОБЬЕВ
(г. Тула).



```

10 REM ПОИСК МИНИМУМА ФУНКЦИИ НА ОТРЕЗКЕ
   [A,B] МЕТОДОМ ЗОЛОТОГО СЕЧЕНИЯ
20 DEFN Y(X)=...:REM ВИД ФУНКЦИИ
30 INPUT "A,B,ТОЧНОСТЬ",A,B,E
40 GOSUB 110:GOSUB 120
50 IF ABS(B-A)<E THEN 90:REM НАЧАЛО ЦИКЛА "ПОКА"
60 IF Y1>Y2 THEN 70:REM АЛЬТЕРНАТИВА:
   B=X2:X2=X1:Y2=Y1:
   GOSUB 110:GOTO 80
70 A=X1:X1=X2:
   Y1=Y2:GOSUB 120
80 GOTO 50:REM КОНЕЦ ЦИКЛА "ПОКА"
90 X=(A+B)/2:PRINT "Y МИН=";FNY(X);" ПРИ X=";X
100 END:REM КОНЕЦ ПРОГРАММЫ, НАЧАЛО ПОДПРОГРАММ
110 X1=.618*A+.382*B:Y1=FNY(X1):RETURN
120 X2=.382*A+.618*B:Y2=FNY(X2):RETURN
    
```

ПРЕДСТАВЛЯЕМ НЕКОТОРЫЕ КНИГИ ПО БЕЙСИКУ, ВЫШЕДШИЕ В СВЕТ В ПОСЛЕДНЕЕ ВРЕМЯ

Жигарев Н. В., Макарова Н. В., Путинцева М. А. **Основы компьютерной грамоты.** Л.: Машиностроение. Ленинградское отделение, 1987. Устройство ЭВМ, Бейсик «Искры 226», применение ЭВМ в народном хозяйстве.

Никс Дж. **Решение производственных задач на Бейсике:** Пер. с англ. М.: Машиностроение, 1987. Программы для технолога металлообрабатывающих производств.

Маркелова Л. Н. **Эксплуатация программ управляемой вычислительной машины «Искра 226».** М.: Машиностроение, 1987. Краткое изложение руководства по программированию, прилагаемого к ППЭВМ «Искра 226».

Баласанян В. Э., Богдюкевич С. В., Шахвердов В. А. **Программирование на микроЭВМ «Искра 226».** М.: Финансы и статистика, 1987. Описание версии Бейсика ППЭВМ «Искра 226» (без графических операторов).

Эберт К., Эдерер К. **Компьютеры. Применение в химии:** Пер. с нем. М.: Мир, 1988. Описание численных методов, их иллюстрация примерами из химии. Программы написаны на Бейсике ППЭВМ Коммодор CBM 8032.

Банди Б. **Методы оптимизации.** Вводный курс: Пер. с англ. М.: Радио и связь, 1988. Программы на Бейсике IBM PC для оптимизации функций одной и многих переменных.

Дьяконов В. П. **Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ:** Справочник. М.: Наука, 1987. Более 300 программ для микроЭВМ «Электроника ДЗ-28», предлагаемых для решения задач численного анализа, электро- и радиотехники.

Консевски Ч. **Занимательная математика и персональный компьютер:** Пер. с англ. М.: Мир, 1987. Программы на Бейсике задач из теории игр, теории вероятности.

Очков В. Ф., Пухначев Ю. В. **24 этюда на Бейсике.** М.: Финансы и статистика, 1988. Задачи теории игр, численного и текстового анализа, примеры создания локальных банков данных.

Уолш Б. **Программирование на Бейсике:** Пер. с англ. М.: Радио и связь, 1988. Версии Бейсика различной сложности, структурное программирование на Бейсике.