
ИЗДАТЕЛЬСТВО
МИР

La pratique du Fortran

Exercices commentés

Michel Dreyfus

Directeur du Centre de Calcul de l'Institut National d'Astronomie et de Géophysique

Claude Gangloff

Chef d'exploitation de l'Institut National d'Astronomie et de Géophysique

DUNOD

PARIS 1975

М. ДРЕЙФУС
К. ГАНГЛОФ

ПРАКТИКА ПРОГРАММИРОВАНИЯ НА ФОРТРАНЕ

УПРАЖНЕНИЯ С КОММЕНТАРИЯМИ

Перевод с французского
В. А. Баяковской и
А. А. Бряндинской

под редакцией
Ю. М. Баяковского

ИЗДАТЕЛЬСТВО «МИР»
МОСКВА 1978

УДК 681.142.2

Для овладения искусством программирования на Фортране недостаточно знать его грамматику. Необходимо уметь рационально использовать все средства языка, а это приходит с практикой и опытом. В предлагаемой книге авторы стремятся на многочисленных и разнообразных задачах передать свой многолетний опыт практической работы с Фортраном. Для каждой задачи они приводят несколько (порой до десятка) решений и комментируют их достоинства и недостатки. Большое внимание уделяется вопросам оптимизации программ.

Книга рассчитана на широкий круг программистов — инженеров и научных сотрудников, применяющих Фортран в своей работе; она будет способствовать улучшению стиля программирования, повышению качества и эффективности программ на Фортране.

Редакция литературы по математическим наукам

Д $\frac{20204\text{-}021}{041(01)\text{-}78}$ 21-78

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА

В первой главе этой книги в качестве примера приводятся 10 различных решений одной простой задачи и дается их количественная оценка для вычислительной машины IBM 360/65. Из сравнения этих решений видно, что ценой вроде бы совсем небольших и довольно очевидных усовершенствований в алгоритме, а также благодаря умелому выбору возможностей языка удается почти в 6 раз уменьшить время работы программы. Этот пример свидетельствует о том, что в стиле программирования кроются значительные резервы повышения эффективности работы вычислительных центров.

Проблема улучшения стиля программирования особенно важна для Фортрана в силу его массового применения. Несмотря на то, что Фортран используется уже 30 лет, вопросы стилистики Фортрана почти не нашли отражения в литературе. Тем больший интерес представляет попытка авторов настоящей книги изложить в форме упражнений свой многолетний опыт программирования на Фортране.

Все программы, которые приводятся в книге, были проверены на машинах. При подготовке русского перевода некоторые из них были опробованы на машине БЭСМ-6 для двух компиляторов БЭСМ-6/Д (компилятор, разработанный в Объединенном институте ядерных исследований, Дубна) и БЭСМ-6/М (компилятор, разработанный в Институте прикладной математики АН СССР, Москва). Во втором компиляторе осуществляется локальная оптимизация, которая во многих случаях дает ощутимый эффект.

Наряду с традиционными в книге рассматриваются применения Фортрана в областях, которые считаются исключительной привилегией других языков, предназначенных для обработки текстов, файлов. Надо сказать, что неудобства, которые при этом возникают, устранены в новом американском стандарте — Фортран 77. В нем предусмотрены циклы с отрицательным шагом, возможность задания массивов с указанием граничных пар индексов, переменные типа CHARACTER, конструкция IF — THEN — ELSE и т. д. В отличие от многих других языков программирования Фортран — развивающийся

язык; в настоящее время предпринимается попытка подготовить следующий стандарт — Фортран 82, чтобы закрепить в нем новые конструкции, проверенные в различных диалектах, и исключить архаизмы.

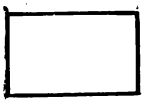
Настоящая книга основывается на распространенном сейчас стандарте Фортран IV, и большинство программ без каких-либо изменений пригодны для решения на любой из существующих машин.

Книга рассчитана на программистов, имеющих некоторый опыт работы на Фортране. Начинающим, по-видимому, полезно прежде познакомиться с книгой Ж.-П. Ламуатье «Упражнения по программированию на Фортране IV», выходящей также в издательстве «Мир».

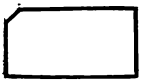
Специально для русского издания авторы любезно прислали ряд исправлений и список опечаток, и мне приятно выразить им свою благодарность.

Ю. Баяковский

ОБОЗНАЧЕНИЯ, ПРИНЯТЫЕ В БЛОК-СХЕМАХ



обработка

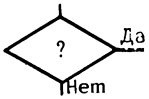


ввод данных с перфокарт

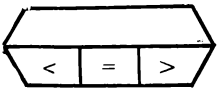
вывод данных на печать

операция с магнитной лентой

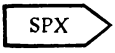
конец обработки



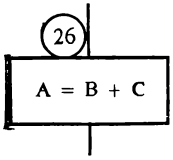
проверка. Внутри ромба стоит вопрос. Если ответ «нет», то выход вниз, если «да» — налево или направо (логический IF)

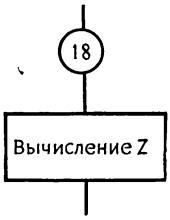


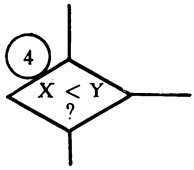
проверка с тремя выходами (арифметический IF)



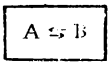
вызов подпрограммы или функции



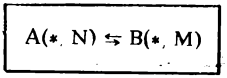




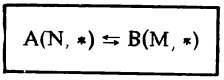
различные способы указания метки инструкции



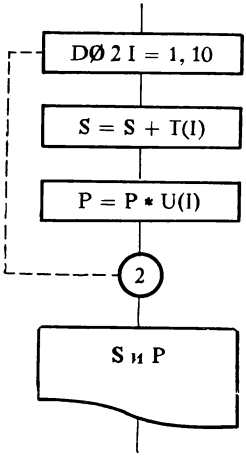
перестановка элементов A и B (простые переменные)



перестановка N-го столбца A с M-ым столбцом B



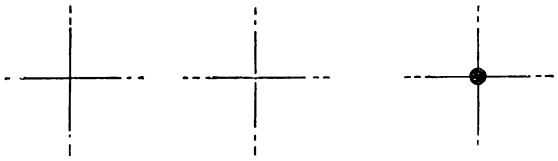
перестановка N-й строки A с M-й строкой B



DØ-циклы обозначены пунктиром, ограничивающим тело цикла, и отдельно стоящим кружком с меткой их конца. Кроме того, шаг цикла не уточняется; он указан в самой инструкции DØ.

Эти соглашения не соответствуют установленному стандарту. Но мы их чаще всего встречаем у самых разных по специальности пользователей вычислительных машин. Мы их приняли здесь, поскольку нам кажется, что, кроме простоты, их большим достоинством являются ясность и однозначность.

Наконец, две линии, встречающиеся под прямым углом, вовсе не обозначают «замыкание» в этой точке; предполагается, что они скрещиваются в пространстве, не пересекаясь. Если мы желаем показать, что они действительно пересекаются, мы обозначим их пересечение жирной точкой:



не пересекаются пересекаются.

Глава 1

ВВЕДЕНИЕ

Как нельзя научиться бегло говорить на языке, познакомившись с ним только по учебнику, так нельзя стать хорошим программистом, не поработав на вычислительной машине. Нет недостатка в учебниках, содержащих описание грамматики и синтаксиса Фортрана, и они действительно необходимы для усвоения основных правил. Затем, после первых мук ученичества приходит мастерство. Не слишком важно, написана ли программа с изяществом, но желательно оптимизировать ее, чтобы при эксплуатации она была экономичной в смысле вычислительного времени и места в памяти. Именно в этот момент возникает потребность узнать возможности каждой инструкции.

И поэтому мы попытались на примерах очень разных программ показать не только то, что надо делать, но прежде всего то, чего следует избегать. Чтобы уменьшить скуку, которую могут вызвать слишком схоластические упражнения, мы стремились разнообразить темы, привлекая там, где было возможно, элемент фантазии, но не жертвуя при этом строгостью рассуждений.

Таким образом, в нашей книге не будет отдельной главы, посвященной операторам IF, главы, посвященной DØ-циклам, еще одной — подпрограммам, ... Однако за двумя исключениями: мы начнем с напоминания о вводе-выводе и, кроме того, подробному анализу малоизвестных и часто недооцениваемых возможностей Фортрана предпошлем главу, посвященную манипуляциям с текстами. Различные затронутые темы были выбраны с учетом их практической полезности и частоты применений.

Все без исключения примеры и упражнения, вошедшие в эту книгу, были проверены на вычислительной машине. С этой целью использовались следующие машины:

CII Mitra 15,
CONTROL DATA 6600,

IBM 360/65,
TELEMECANIQUE T1600,
UNIVAC 1106.

Пользуясь случаем, авторы выражают искреннюю благодарность тем фирмам, которые дали возможность воспользоваться своим оборудованием.

Чтобы предложенные решения носили в какой-то мере универсальный характер, несмотря на видоизменения, которые появились в Фортране при его реализации разными разработчиками, мы решили принять за образец язык Фортран IV, признавая в нем наряду с другими следующие характеристики:

- имена переменных, состоящих не более чем из 6 литер,
- допускается смешение типов в арифметических выражениях,
- отсутствует рекурсивность на уровне вызова подпрограмм,
- в DØ-циклах допускается лишь положительный «шаг», причем если начальное значение параметра больше его конечного значения, цикл тем не менее выполняется один раз,
- индексные выражения ограничены классической формой: c, v, c * v, c ± v, c * v ± c′,
- допускаются переменные типов INTEGER, REAL, LØGICAL, DØUBLE PRECISIØN и CØMPLEX,
- предполагается совместимость с Фортраном II на уровне инструкций READ, PRINT и PUNCH, которые будут систематически использоваться из-за их педагогической наглядности.

Звездочками отмечены трудные упражнения. Начинающие могут сразу же приниматься за упражнения, отмеченные одной звездочкой. Приобретя некоторую уверенность, они смогут взяться за упражнения с двумя звездочками, наконец, опытным программистам, даже виртуозам, стоит испытать себя на упражнениях с тремя звездочками.

Подробность приведенных объяснений при каждом решении уменьшается с возрастанием трудности упражнения. Действительно, если начинающему программисту, мало знакомому с особенностями той или иной инструкции, необходимо обстоятельно разъяснять механизм, то опытный программист не нуждается в подробностях, и только тонкости логического рассуждения, приведшего к выбору того или иного метода решения, важны для него. Поэтому-то мы и не советуем начинающим программистам слишком рано приниматься за трудные упражнения. В трудных упражнениях мы пытались при-

близиться к реальности конкретных приложений, но все же были вынуждены внести в них некоторые упрощения, чтобы более выпукло показать достоинства предложенного метода, не похоронив их под грудой излишних деталей.

В ряде случаев при одном и том же условии задачи мы приводим несколько решений в порядке возрастания их эффективности и соответственно трудности. Количество звездочек тогда увеличивается.

Главным образом в простых упражнениях для дополнительных пояснений используются блок-схемы. Принятые нами соглашения подробно описаны на стр. 7—8.

Некоторые другие соглашения

Для обозначения элементов массивов примем следующие соглашения:

Пусть A (10, 30) — массив. Посредством A(*, 2) обозначим второй столбец массива. Аналогично, через A(7, *) обозначим седьмую строку массива A. Конечно, речь не идет о том, что мы будем писать это в программе, мы используем это соглашение лишь в тексте условия или решения.

Буква О будет представлена литерой «Ø», а нуль просто литерой «О». Для представления пробелов в текстовых константах используется маленькая «крышка» внизу строки: 1Н␣ следует читать «1Н пробел».

Треугольник Паскаля

Чтобы проиллюстрировать принятый в нашей книге метод изложения, рассмотрим простое упражнение, которое можно найти во всех учебниках по программированию, независимо от изучаемого языка, а именно треугольник Паскаля. Так называется простая схема, которая дает коэффициенты разложения бинома Ньютона $(a+b)^n$. Требуется получить треугольную таблицу:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
. . . . . .
```

Будем вычислять, а затем печатать такую таблицу до 16 порядка (принимая во внимание число цифр в коэффициентах и

размер строки печатающего устройства, мы стремились получить наглядно представленный результат).

Возможны несколько решений. Не претендуя на исчерпывающие обсуждения всех возможных методов, мы начнем с самого элементарного и покажем, как при решении даже такой простой задачи можно использовать столь различные методы.

1-е решение

Вычислим все коэффициенты ручным способом или с помощью карманного калькулятора. Работа невелика и не требует никаких операций, кроме сложений, так как любой коэффициент получается из коэффициентов предыдущей строки с помощью алгоритма, представленного следующей схемой:

```
1 ... X  X  X + X  X ...
                ↓
1 ... X  X  X   X
```

По каждой строке значений перфорируется одна карта, т. е. всего 17 карт; составляется программа из 8 инструкций, которая вводит, а затем печатает каждую карту последовательно. Хотя это решение имеет два достоинства (оно простое и дает результаты, точность которых зависит лишь от умения программиста складывать), мы отвергаем его как ребяческое и не имеющее абсолютно никакой педагогической ценности.

2-е решение

Первая серьезная мысль, которая приходит на ум, это взять квадратный массив из 17×17 элементов и присвоить им нулевые значения, поскольку ни в коем случае не следует делать предположений относительно начальных значений переменных, которым никогда не были присвоены никакие числовые значения ни инструкцией READ, ни оператором DATA, ни знаком «=». Затем занесем «1» в верхний левый угол массива и будем применять, начиная со второй строки, соотношение:

$$N_{I,\,j} = N_{I-1,\,j-1} + N_{I-1,\,j}.$$

Первому элементу первой строки необходимо отдельно присвоить начальное значение, так как в этом случае при $i = 1$ индекс $(i - 1)$ равен 0 и соответствующий элемент массива не определен. По той же причине необходимо отдельно обрабатывать первый элемент каждой строки, так как индекс $j - 1$ в этих случаях равен 0. Можно найти два очень близких решения в соответствии с блок-схемами на рис. 1 и 2, кото-

рые мы назовем «решение 2А» и «решение 2В». Они практически эквивалентны, и соответствующие программы (печать исключена и будет рассмотрена позднее) воспроизведены на рис. 3 и 4.

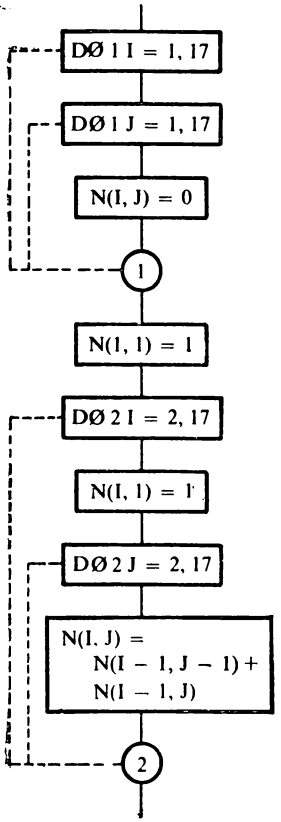


Рис. 1

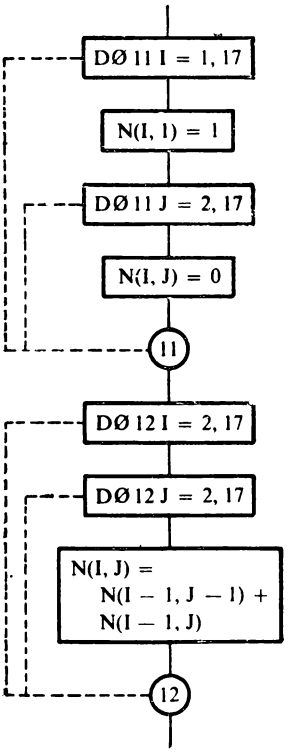


Рис. 2

Заметим, что в обоих случаях имеются две группы из двух вложенных DØ-циклов. Начальные установки немного различаются, но в обоих случаях часть времени уходит на суммирование нулевых элементов (верхний треугольник).

3-е решение

Однако число ненулевых элементов каждой строки равно ее порядковому номеру (1 элемент в первой строке, 2 элемента во второй, ..., 17 в последней). Мы решаем сократить число сложений, задавая в последнем цикле переменный предел, что и сделано в программе на рис. 5.

```
C**                                              РЕШЕНИЕ N 2
      DO 1 I=1,17
      DO 1 J=1,17
1     N(I,J)=0
      N(1,1)=1
      DO 2 I=2,17
      N(I,1)=1
      DO 2 J=2,17
2     N(I,J)=N(I-1,J-1)+N(I-1,J)
```

Рис. 3

```
      DO 11 I=1,17
      N(I,1)=1
      DO 11 J=2,17
11    N(I,J)=0
      DO 12 I=2,17
      DO 12 J=2,17
12    N(I,J)=N(I-1,J-1)+N(I-1,J)
```

Рис. 4

```
C**                                              РЕШЕНИЕ N 3
      DO 21 I=1,17
      N(I,1)=1
      DO 21 J=2,17
21    N(I,J)=0
      DO 22 I=2,17
      DO 22 J=2,I
22    N(I,J)=N(I-1,J-1)+N(I-1,J)
```

Рис. 5

4-е решение

Подумав еще немного, мы обнаружим, что пределы нескольких циклов одинаковы, и перестроим блок-схему так, чтобы все вычисления были сгруппированы только в два цикла. Теперь установка начальных значений производится уже не предварительно, а постепенно, и мы избавляемся от бесполезных вычислений, проверяя индексы I и J каждого элемента. Блок-схема воспроизведена на рис. 6, а программа — на рис. 7. Можно было также написать

```
      | . . . . . .
      | IF (I.LE.J) GØTØ2
      | N(I, J) = 0
      | GØTØ1
2     | N(I, J) = N(I — 1, J — 1) + N(I — 1, J)
1     | CØNTINUE
```

но эта запись более громоздка, и программа выполняется медленнее. Если же написать

```
      . . . . . .
      IF (I.LE.J) N(I, J) = N(I - 1, J - 1) + N(I - 1, J)
      IF (I.GT.J) N(I, J) = 0
1     CØNTINUE
```

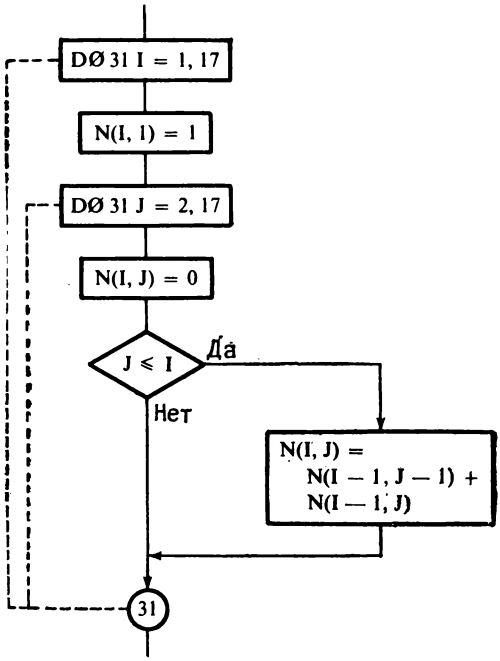


Рис. 6

```
C**                                          РЕШЕНИЕ N 4
      DO 31 I=1,17
      N(I,1)=1
      DO 31 J =2,17
      N(I,J)=0
      IF(J.LE.I) N(I,J)=N(I-1,J-1)+N(I-1,J)
31    CONTINUE
```

Рис. 7

то внешняя элегантность обойдется каждый раз в две систематические проверки, что отнюдь не уменьшает времени выполнения. Столь же неразумно в программе на рис. 7

инвертировать отношение в логическом выражении и писать

```
      | |  . . . . .
      | |  N(I, J) = N(I − 1, J − 1) + N(I − 1, J)
      | |  IF (I.GT.J) N (I, J) = 0
    1 | |  CØNTINUE
```

так как пришлось бы систематически вычислять простое выражение, требующее, однако, вычисления индексов, и исключить при этом лишь простое присваивание нулевых значений.

Печать

В каждой из предыдущих программ можно написать

PRINT 100, ((N(I, J), J = 1,17), I = 1,17)

При этом будут напечатаны все элементы массива, даже нулевые; или можно написать

```
      | |  DØ 5 I = 1,17
    5 | |  PRINT 100, (N(I, J), J = 1,17)
  100 | |  FØRMAT (17I6)
```

что даст тот же результат. Но не следует писать

```
| | PRINT 100, N
```

поскольку тогда печатался бы транспонированный (т. е. повернутый на 90°) массив, где каждая строка на бумаге соответствовала бы столбцу массива.

Чтобы напечатать лишь ненулевые элементы, можно написать

```
| | PRINT 100, ((N(I, J), J = 1, I), I = 1, 17)
```

Действительно, будут печататься только ненулевые элементы, но их расположение оставляет желать лучшего. Каждая строка в массиве имеет разное число ненулевых элементов, оператор FØRMAT предусматривает постоянное число элементов в печатаемой строке. Так как имеется *лишь одна* инструкция вывода, начало повторного просмотра формата не совпадает с переходом к новой строке массива. Результатом будет таблица из 9 строк по 17 значений в каждой, в которой трудно будет увидеть наш треугольник.

Чтобы получить привычное треугольное расположение, нужна *не одна* инструкция печати, а *столько, сколько строк в треугольной таблице.* Таким образом, мы опять приходим к использованию явного DØ-цикла:

```
      | |  DØ 6 I = 1, 17
    6 | |  PRINT 100, (N(I, J), J = 1, I)
```

5-е решение

Но если нулевые элементы не печатаются, то зачем их вычислять? Ограничившись вычислением лишь ненулевых членов, мы приходим к программе на рис. 8. К сожалению, это решение неудачно. В самом деле, в каждой новой строке вычисляется один дополнительный элемент, и в используемом алгоритме неявно предполагается, что все элементы предыдущей строки существуют, включая и нули в конце строки. Однако неизвестно, что стоит справа от единицы. Не следует забывать, что невычисленные элементы имеют обычно неопределенные значения (об этом мы уже говорили выше). Можно было бы всем элементам массива присвоить нулевые значения, но тогда бы мы вновь вернулись к тяжеловесности, свойственной предыдущим решениям.

```
C**                                    РЕШЕНИЕ N 5
      N(1,1)=1
      DO 41 I=2,17
      N(I,1)=1
      DO 41 J=2,I
41    N(I,J)=N(I-1,J-1)+N(I-1,J)
```

Рис. 8

Лучше было бы присвоить начальные значения с помощью оператора DATA:

```
DATA N/289*0/
```

6-е решение

Здесь мы учитываем тот факт, что значение последнего элемента каждой строки известно: оно равно «1», и мы при-

```
C       ТРЕУГОЛЬНИК ПАСКАЛЯ- РЕШЕНИЕ N 6
C
        N(1,1)=1
        DO 51 I=2,17
        N(I,1)=1
        N(I,I)=1
        N(I-1,I)=0
        L=I-1
        DO 51 J=2,L
51      N(I,J)=N(L,J-1)+N(L,J)
```

Рис. 9

сваиваем это значение вначале, а затем вычисляем в i-й строке элементы со 2-го до (i — 1)-го, см. рис. 9. Заметим, что там всего один DØ-цикл. В самом деле, при I = 1 и I = 2 не надо вычислять никаких элементов, потому что единственные

имеющиеся элементы — это одна или две единицы. Если DØ-цикл один, то в этих случаях его конечный параметр будет меньше своего начального значения, но известно, что по правилам Фортрана цикл все же выполняется один раз. Тогда вычисленный результат будет ошибочным.

Этот внутренний цикл должен, таким образом, выполняться для всех значений J от 2 до I — 1, за исключением I = 2. Следовательно, надо выходить из цикла, когда J принимает такое же значение, что и I. Но такая проверка позволит также исключить выполнение цикла при I = 2, так как это первое значение, которое принимает J. Помещая проверку в начале цикла и используя условие «равно», получаем правильную программу, которую невозможно было бы написать с одним простым DØ-циклом в этом месте [1]).

7-е решение

Можно также «смириться» и вычислять лишний элемент, не являющийся строго необходимым: первый нуль справа от

```
C**                                          РЕШЕНИЕ N  7
      N(1,1)=1
      DO 61 I=2,17
      N(I,1)=1
      N(I-1,I)=0
      DO 61 J=3,I
61    N(I,J-1)=N(I-1,J-2)+N(I-1,J-1)
```

Рис. 10

```
C**                                          РЕШЕНИЕ N  8
      M(1)=1
      PRINT 100,M(1)
      DO 74 I=2,17
      M(I)=0
      DO 72 J=2,I
72    NN(J)=M(J-1)+M(J)
      DO 73 J=2,I
73    M(J)=NN(J)
74    CONTINUE
```

Рис. 11

последней единицы. Но чтобы не выйти за пределы массива в последней строке, надо вычислить (или скорее «вставить») нуль в (i — 1)-ю строку во время вычисления i-й строки. Та-

[1]) В программе на рис. 9 обошлись без такой проверки, поскольку вычисленное значение N(2, 2) равно единице и совпадает со значением, присвоенным этому элементу ранее. — *Прим. ред.*

ким образом, придем к решению, представленному на рис. 10. Обратите внимание, что мы избежали отдельного вычисления I — 1, конечного значения параметра DØ-цикла по J, начав этот цикл с 3 и используя J — 1 в индексном выражении, там где раньше было J. Но это не слишком удачная мысль, так как индексное выражение вычисляется дольше, чем простой индекс, который нужно только взять из памяти [1]).

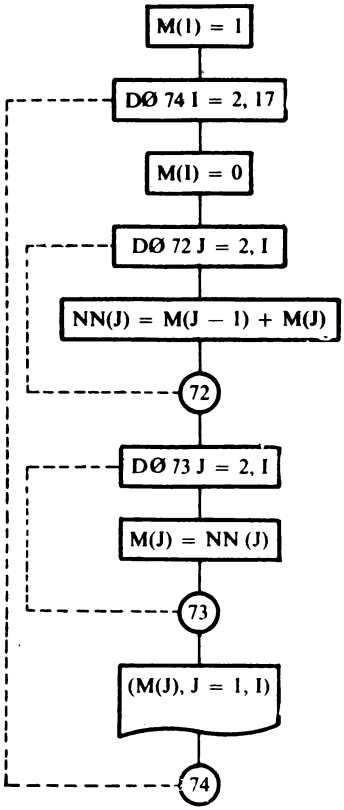


Рис. 12

8-е решение

Теперь мы рассмотрим три решения, в которых используются одномерные массивы.

Очевидное преимущество использования одномерного массива выражается в уменьшении объема требуемой памяти. В этом решении необходимо два массива по 17 элементов в каждом, т. е. 34 слова памяти, по сравнению с 289 словами в предыдущих решениях. Теперь мы не будем сначала вычислять, а потом печатать. Очевидно, это невозможно, потому что этих двух массивов недостаточно, чтобы хранить целиком треугольник. Каждый раз, когда вычислена одна строка, она тотчас же печатается. На рис. 11 мы видим инструкцию, которая печатает только первый элемент, поскольку, как это часто бывает в программировании, метод вычисления нуждается в «стартере», т. е. в начальных установках, определяющих конкретные условия. Далее до самого конца работает тот же алгоритм, что и в предыдущих случаях.

---

[1]) Массив должен быть описан в инструкции INTEGER N(17, 17). Тогда для N(I, J) вычисляется индексное выражение 17 * (J — 1) + (I — — 1) = 17 * J + I — 18, а для N(I, J — 1) — выражение 17 * (J — 2) + + (I — 1) = 17 * J + I — 35. Как видим, второе выражение ничуть не сложнее первого. Поэтому высказанное в тексте предположение справедливо лишь для очень «убогих» трансляторов. — *Прим. ред.*

Но мы знаем, что для получения новой строки достаточно иметь предыдущую. Поэтому мы сохраняем лишь предыдущую строку (массив M) и, исходя из него, вычисляем текущую строку (массив NN). На блок-схеме (рис. 12) подробно показаны инструкции и то, что перед печатью массив NN переносится в M, т. е. вычисленная строка занимает место предыдущей.

Это решение может показаться сложным, но мы увидим, что оно очень эффективно в первую очередь благодаря использованию массива с одним индексом (под словом «эффективное» мы понимаем «требующее немного вычислительного времени», но, разумеется, обеспечивающее во всех случаях необходимую точность результатов).

9-е решение

Приложив немного стараний, можно обойтись без одного из двух массивов. Однако это все же не так просто, как может показаться на первый взгляд. Пунктуальное применение

```
C**                                    РЕШЕНИЕ  N  9
      M(1)=1
      PRINT 100,M(1)
      DO 81 I=2,17
      K=I+2
      M(I)=0
      DO 82 J=2,I
      L=K-J
82    M(L)=M(L)+M(L-1)
81    CONTINUE
```

Рис. 13

алгоритма влечет за собой изменение элемента на слишком ранней стадии, до того как он станет ненужен (для вычисления i-го элемента надо располагать (i — 1)-м, но это уже не элемент предыдущей строки). Чтобы обойти эту трудность, достаточно начать вычисления справа, потому что в каждой строке на один элемент больше, чем в предыдущей. Таким образом приходим к программе на рис. 13. Мы знаем, что в DØ-циклах обычно не допускаются «шаги в обратную сторону». Вычисляя дополнительно L = K — J, мы преодолеваем эту новую трудность и, хотя остается еще довольно много инструкций [1]), видим, что получается выигрыш во времени по

[1] Если же допускается отрицательный шаг, то программу можно упростить, исключив инструкции K = I + 2 и L = K — J и заменив заголовок внутреннего цикла на DO 82 L = K — 2, K — I, —1. Например, Фортран 77 позволяет вести такие изменения. — *Прим. ред.*

сравнению с предыдущими решениями из-за устранения пересылки одного массива в другой при переходе к следующей строке.

10-е решение

Теперь попытаемся сохранить все ненулевые значения треугольника в одномерном массиве из 153 элементов

$$(1+2+3+4+\ \ldots\ +17=153).$$

Получим

1	1 1	1 2 1	1 3 3 1	1 4 6 4 1	1 ...	треугольник
1	2 3	4 5 6	7 8 9 10	11 12 13 14 15	16 ...	индекс

Поскольку теперь уже нет нулей, то во избежание проверок, которые удлинили бы вычисления, начинаем с заполнения всего массива N единицами (см. рис. 14) в цикле DØ3. Затем перейдем к циклу DØ1 K, где K принимает последовательно все значения, соответствующие числу отличных от единицы элементов в каждой строке, начиная с третьей.

Значение J, вначале равное 3 (индексу первого отличного от единицы элемента), вычисляется по формуле J + K + 1 и при каждом повторении цикла заменяет прежнее значение J. Таким образом, индекс последнего элемента, отличного от единицы, равен индексу первого + K (число отличных от единицы элементов) — 1, т. е. L = J + K — 1. Значения J и L служат начальным и конечным значениями параметра цикла DØ1 I, в котором вычисляются N(I), соответствующие одной строке.

Для печати массива используется очень похожая программа. На этот раз надо печатать все элементы, а не только отличные от единицы. Таким образом, K изменяется от 1 до 17, и в зависимости от K вычисляются значения J и L.

Мы привели это решение как занимательное, хотя и бесполезное из-за сложности вычисления индекса и как следствие далеко не простого доступа к элементам массива.

Сравнительные характеристики 10 решений

Чтобы можно было сравнить эффективность этих решений, мы «прокрутили» каждое из них по 500 раз во внешнем DØ-цикле, до и после которого написали оператор CALL TIME (нестандартная подпрограмма, позволяющая определить время, прошедшее с предыдущего вызова). Печать была

```
C       ТРЕУГОЛЬНИК ПАСКАЛЯ РЕШЕНИЕ С ОДНОМЕРНЫМ
C       МАССИВОМ
C
        DIMENSION N(153)
        DO 3 I=1,153
3       N(I)=1
        J=3
        DO 1 K=1,15
        J=J+K+1
        L=J+K-1
        DO 1 I=J,L
1       N(I)=N(I-K-2)+N(I-K-1)
        J=1
        DO 4 K=1,17
        J=J+K-1
        L=J+K-1
4       PRINT 100, (N(I),I=J,L)
100     FORMAT (1716)
        STOP
        END
```

```
 1
 1  1
 1  2  1
 1  3  3  1
 1  4  6  4  1
 1  5 10 10  5  1
 1  6 15 20 15  6  1
 1  7 21 35 35 21  7  1
 1  8 28 56 70 56 28  8  1
 1  9 36 84 126 126 84 36 9  1
 1 10 45 120 210 252 210 120 45 10 1
 1 11 55 165 330 462 462 330 165 55 11 1
 1 12 66 220 495 792 924 792 495 220 66 12 1
 1 13 78 286 715 1287 1716 1716 1287 715 286 78 13 1
 1 14 91 364 1001 2002 3003 3432 3003 2002 1001 364 91 14 1
 1 15 105 455 1365 3003 5005 6435 6435 5005 3003 1365 455 105 15 1
 1 16 120 560 1820 4368 8008 11440 12870 11440 8008 4368 1820 560 120 16 1
```

Рис. 14

упразднена, чтобы измерить только время вычисления. В результате получили следующую таблицу:

Решение	2	3	4	5	6	7	8	9	10
Время	5,8	3,9	4,0	2,2	2,1	2,0	1,4	1,2	1,2

(решения 2A и 2B дали очень близкие времена, и мы взяли их среднее; что же касается решения 1, мы не сочли полезной его реализацию).

Время дано в секундах и относится к вычислительной машине IBM 360/65 [1]).

[1]) Надо полагать, что многие читатели не преминут повторить эти исследования на машинах, имеющихся в их распоряжении. Мы приведем здесь результаты, полученные при тех же условиях на ЭВМ БЭСМ-6 для двух разных компиляторов.

Номер решения	2	3	4	5	6	7	8	9	10
БЭСМ-6/Д	12,1	10,0	10,1	4,2	4,0	3,4	3,6	2,3	4,3
БЭСМ-6/М	1,4	1,5	2,4	0,7	1,0	0,7	1,1	1,3/0,8*	1,0

(Результат, отмеченный *, получен после внесения в программу изменений, предложенных в предыдущей сноске.) У нас нет здесь места для анализа этих результатов. Но обратите внимание: сколь сильно эффективность использования ЭВМ может зависеть от качества транслятора и квалификации программиста. — *Прим. ред.*

глава 2

ВВОД-ВЫВОД

2.1 НАПОМИНАНИЕ ОБЩИХ ПОНЯТИЙ

Вспомним три принципа, которые управляют процессом форматного ввода-вывода:

1. список ввода-вывода, если он задан, всегда исчерпывается до конца,

2. повторный просмотр формата сопряжен с переходом к следующей записи,

3. повторный просмотр делается с самого начала формата, если в нем нет группы спецификаций, заключенной в скобки; в противном случае просматривастся самая правая *полная группа.*

Если речь идет о вводе, то в операторе FØRMAT описывается карта. Следующие два примера, снабженные обширным комментарием, при сопоставлении с полученными результатами достаточно полно иллюстрируют практическое применение этих простых правил.

Программа на рис. 15 вводит карты (рис. 16) и печатает результаты (рис. 17). Во втором примере рассматривается только печать. С помощью программы, листинг которой приведен на рис. 18, получены результаты, представленные на рис. 19.

2.2 РИСОВАНИЕ КРИВОЙ НА АЦПУ **

(*решение на стр.* 76)

Требуется на алфавитно-цифровом печатающем устройстве (АЦПУ) нарисовать точками кривую. Координаты точек задаются в массиве T(200, 2), причем столбец T(*, 1) содержит значения абсцисс, а столбец T(*, 2) — значения соответствующих ординат. Предполагается, что кривая не должна деформироваться, и поэтому следует учитывать горизонтальное расстояние между литерами ($^1/_{10}$ дюйма) и вертикальное рас-

```
C       ПРИМЕРЫ ВВОДА МАССИВОВ
C
C
C                       ОДНОМЕРНЫЕ МАССИВЫ
C
        DIMENSION T(20)
C
C       МАКСИМАЛЬНОЕ ЧИСЛО ЭЛЕМЕНТОВ ВВОДА РАВНО: 20
C       НА КАРТЕ ОДИН ЭЛЕМЕНТ
C
        READ 1, T
1       FORMAT (F5.1)
        PRINT 2, T
2       FORMAT (//20F6.1)
C
C       НА КАРТЕ НЕ БОЛЕЕ 16 ПОДРЯД СТОЯЩИХ ЭЛЕМЕНТОВ
        READ 3, T
3       FORMAT (16F5.1)
        PRINT 2, T
C
C
C       НЕОБХОДИМО ВВЕСТИ ТОЛЬКО  'N' ПЕРВЫХ ЭЛЕМЕНТОВ (N<ИЛИ=20)
C       ВНАЧАЛЕ ЛЕЖИТ КАРТА, СОДЕРЖАЩАЯ N
        READ 4, N, (T(I), I=1, N)
4       FORMAT (I2 / (F5.1) )
        PRINT 5, N, (T(I), I=1,N)
5       FORMAT (/I3, 20F6.1)
C
C       ЗАРАНЕЕ НЕИЗВЕСТНО ЧИСЛО ВВОДИМЫХ ЭЛЕМЕНТОВ, НО ПОСЛЕДНЯЯ ИЗ КАРТ
C       В КОЛОНКЕ 80 СОДЕРЖИТ "9"
        DO 6 I=1,20
        READ 7, A, J
7       FORMAT (F5.1, 74X I1)
        IF ( J.EQ. 9 ) GO TO 8
6       T(I) = A
8       PRINT 2, (T(K), K=1, I)
C
C       НЕОБХОДИМО ВВЕСТИ ТОЛЬКО ТЕ ЧИСЛА, КОТОРЫМ ПРЕДШЕСТВУЕТ НЕ НУЛЕВОЙ
C       ИНДЕКС. НА КАРТЕ РАСПОЛАГАЕТСЯ ОДНО ЧИСЛО И ЕГО ИНДЕКС. ПРИЗНАКОМ
C       КОНЦА ВВОДА СЛУЖИТ 0
        DO 9 I=1,20
9       T(I) = 0
10      READ 11, K, A
11      FORMAT (I2, F5.1)
        IF ( K .EQ. 0 ) GO TO 12
        T(K) = A
        GO TO 10
12      PRINT 2, T
C
C       ЗАДАЧА ПРЕЖНЯЯ, НО НА КАРТЕ 10 ЧИСЕЛ
        DIMENSION L(10), B(10)
        DO 13 I=1,20
13      T(I) = 0
        DO 14 I=1,2
        READ 15, (L(K), B(K), K=1,10)
15      FORMAT (10(I2, F5.1))
        DO 14 J=1,10
        IF (L(J) .EQ. 0) GO TO 16
        K = L(J)
14      T(K) = B(J)
16      PRINT 2, T
C
        STOP
        END
```

Рис. 15

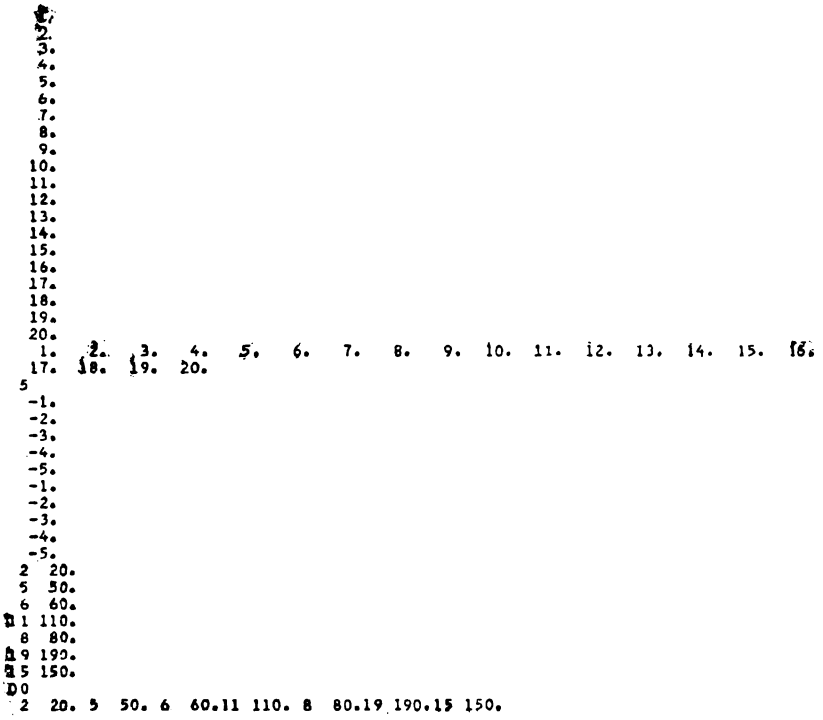

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16.
17. 18. 19. 20.
-1.
-2.
-3.
-4.
-5.
2 20.
5 50.
6 60.
11 110.
8 80.
19 190.
15 150.
2 20. 5 50. 6 60. 11 110. 8 80. 19 190. 15 150.

Рис. 16

глава 3

НЕСКОЛЬКО КЛАССИЧЕСКИХ ЗАДАЧ

3.1. ПРОСТЫЕ ЧИСЛА: САМОЕ ПРОСТОЕ РЕШЕНИЕ *

(*решение на стр.* 87)

Требуется найти и занести в одномерный массив 500 первых простых чисел. Прежде всего предположим, что известны три простых числа: 1, 2 и 3. Известно также, что все последующие простые числа нечетные. Поэтому будем получать число, прибавляя 2 к предыдущему, и затем будем делить это число на все ранее найденные простые числа, включая 3. Если в какой-то момент остаток от деления окажется равным нулю, то проверяемое число не простое. В этом случае к нему прибавляется 2 и процесс повторяется.

Если ни одно из делений не дало нулевого остатка, то это число простое, и следующее проверяемое число получается тем же способом.

Необходимо напечатать полученный массив по 10 чисел в строке.

3.2. ПРОСТЫЕ ЧИСЛА: ПОИСК ЭФФЕКТИВНОГО РЕШЕНИЯ *

(*решение на стр.* 89)

Условие задачи остается прежним, но теперь требуется найти способ, позволяющий уменьшить количество проверок на делимость. В связи с этим заметим, что можно прекратить проверки после деления на все простые числа, меньшие корня квадратного из исследуемого числа.

3.3. ПРОСТЫЕ ЧИСЛА: РЕШЕТО ЭРАТОСФЕНА **

(*решение на стр.* 91)

Этот отличный от предыдущего метод позволяет найти все простые числа, меньшие некоторого заданного числа, в нашем случае 3000. Нам потребуется массив из 3000 слов, который вначале содержит последовательность целых чисел от 1 до 3000 (T(I) = I).

Начнем с 2 и исключим из массива все числа, кратные 2, полагая их равными 0. Затем перейдем к следующему за 2

Рис. 23

```
 1.0   2.0   3.0   4.0   5.0   6.0   7.0   8.0   9.0  10.0  11.0  12.0  13.0  14.0  15.0  16.0  17.0  18.0  19.0  20.0

 1.0   2.0   3.0   4.0   5.0   6.0   7.0   8.0   9.0  10.0  11.0  12.0  13.0  14.0  15.0  16.0  17.0  18.0  19.0  20.0
5  -1.0  -2.0  -3.0  -4.0  -5.0

-1.0  -2.0  -3.0  -4.0  -5.0

 0.0  20.0   0.0   0.0  50.0  60.0   0.0  80.0   0.0   0.0 110.0   0.0   0.0   0.0 150.0   0.0   0.0   0.0 190.0   0.0

 0.0  20.0   0.0   0.0  50.0  60.0   0.0  80.0   0.0   0.0 110.0   0.0   0.0   0.0 150.0   0.0   0.0   0.0 190.0   0.0
```

Рис. 17

```
C     ПРИМЕРЫ ПЕЧАТИ МАССИВОВ
C
      DIMENSION T(5,6), A(5,5), X(5), FORM(3)
      DATA FORM /4H(    , 4H F6., 2HO)/
      DATA X/4H(1X , 4H(9X , 4H(18X, 4H(26X, 4H(34X/
      EQUIVALENCE (T(1,1),A(1,1))
C
C
C     ФОРМИРОВАНИЕ МАССИВА В ПАМЯТИ
      DO 1 I=1,5
      DO 1 J=1,6
1     T(I,J) = 10*I+J
C
C     ЧТОБЫ НАЧАТЬ С НОВОЙ СТРАНИЦЫ
      PRINT 13
13    FORMAT (1H1)
C
C
C     ЭТИ ТРИ ПРИМЕРА ПЕЧАТАЮТСЯ НА ОДНОЙ СТРОКЕ
10    FORMAT (//1X 30F4.0)
C     СПИСОК СОДЕРЖИТ ТОЛЬКО ИМЯ МАССИВА
      PRINT 10,T
C     В НЕЯВНОМ ЦИКЛЕ БЫСТРЕЕ ИЗМЕНЯЕТСЯ ИНДЕКС I
      PRINT 10, ((T(I,J), I=1,5), J=1,6)
C     В НЕЯВНОМ ЦИКЛЕ БЫСТРЕЕ ИЗМЕНЯЕТСЯ ИНДЕКС J
      PRINT 10, ((T(I,J), J=1,6), I=1,5)
C
C
C     ПЕЧАТАЕТСЯ ТАБЛИЦА ИЗ 5 СТРОК ПО 6 СТОЛБЦОВ
      PRINT 11, ((T(I,J), J=1,6), I=1,5)
11    FORMAT (1HO/(6F6.0) )
C
C     ТО ЖЕ САМОЕ, НО ПЕРЕД КАЖДЫМ ЧИСЛОМ ПЕЧАТАЕТСЯ ИМЯ МАССИВА С ИНДЕКСАМИ
      PRINT 99
      PRINT 12, ((I, J, T(I,J), J=1,6), I=1,5)
12    FORMAT  (1X 6(2HT( I1, 1H, I1, 3H) = F6.0, 2X) )
C     ЭТОТ ФОРМАТ НЕ СОВСЕМ КОРРЕКТЕН ПРАВИЛЬНО ТАК
      PRINT 99
99    FORMAT (//)
      PRINT 17, ((I, J, T(I,J), J=1,6), I=1,5)
17    FORMAT ((1X 6(2HT( I1, 1H, I1, 3H) = F6.0, 2X) ))
C
C
C     ТЕПЕРЬ ПЕЧАТАЕТСЯ ТРАНСПОНИРОВАННЫЙ МАССИВ
      PRINT 19, T
19    FORMAT (1HO/(5F6.0))
C
C
C     ПЕЧАТЬ ДИАГОНАЛИ КВАДРАТНОЙ МАТРИЦЫ НА ОДНОЙ СТРОКЕ
      PRINT 99
      PRINT 14, (A(I,I), I=1,5)
14    FORMAT (1X 5(F6.0, 2X) )
C
C     ПЕЧАТЬ ДИАГОНАЛИ КВАДРАТНОЙ МАТРИЦЫ В ОДИН СТОЛБЕЦ
      PRINT 99
      PRINT 15, (A(I,I), I=1,5)
15    FORMAT (1X F6.0)
C
C     ПЕЧАТЬ ДИАГОНАЛИ КВАДРАТНОЙ МАТРИЦЫ ПО ДИАГОНАЛИ
      PRINT 99
      DO 16 I=1,5
      FORM(1) = X(I)
16    PRINT FORM, A(I,I)
      STOP
      END
```

Рис. 18

стояние между ними $^1/_6$ дюйма) [1]). Для вывода кривой удобно выделить поле размером в 60 строк по 100 позиций в строке.

Ось Y маркируется в каждой строке. В заголовке поля печатаются предельные значения и шаг для оси абсцисс. Кривая заключается в рамку, как показано на рис. 20.

Запрещается печатать каждую строку кривой по нескольку раз (т. е. при рисовании кривой нельзя пользоваться литерой «+» для отмены перехода к следующей строке).

2.3. ПЕРЕПИСЬ С КАРТ НА ЛЕНТУ *

(*решение на стр.* 78)

Требуется записать на магнитную ленту колоду перфокарт, содержащих произвольную информацию. Число карт может быть любым. После того как введена и записана последняя карта, лента читается с начала и печатаются образы тех карт, в которых имеется Z в колонке 23 или X в колонке 70. Каждая запись на ленте по конфигурации соответствует карте.

2.4. ПЕРЕПИСЬ С КАРТ НА ЛЕНТУ С ДОПОЛНИТЕЛЬНОЙ ОБРАБОТКОЙ *

(*решение на стр.* 79)

Требуется записать на ленту колоду из 1000 перфокарт. Но по некоторым причинам (в подробности мы вдаваться не будем) желательно, чтобы записи на ленте содержали по

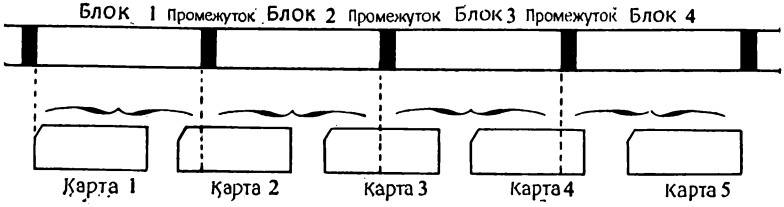


Рис. 21

100 литер. Тогда первая запись на ленте формируется из первой карты и 20 первых литер второй карты и т. д., как показано на рис. 21. Затем при повторном чтении ленты, необходимо напечатать все те записи, которые *предшествуют* записям, содержащим 2 в позиции 98.

---

[1]) В метрической системе эти расстояния равны соответственно 2,54 и 4,23 мм. — *Прим. ред.*

2.5. ДЛЯ ПЕЧАТИ ЗАГОЛОВКА **

(*решение на стр.* 81)

Приятно, когда можно написать большие буквы, такие, как на рис. 22. Поэтому-то и требуется написать программу, единственным аргументом которой является массив из 12 элементов. Элементы массива содержат последовательные слева

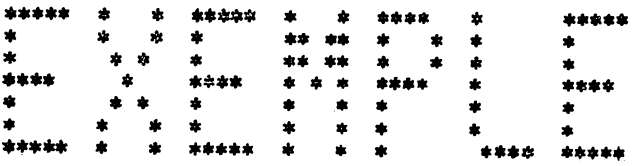


Рис. 22

направо буквы печатаемого слова. Свободные правые позиции заполняются пробелами. К подпрограмме можно обратиться, например, так:

```
      DIMENSIØN MACHIN(12)
      DATA MACHIN/1HE, 1HX, 1HE, 1HM, 1HP, 1HL,
     1                    1HE, 1H^, 1H^, 1H^, 1H^, 1H^/
      CALL TITRE(MACHIN)
```

Разумеется, печатать должна сама подпрограмма, которая называется TITRE.

2.6. ПЕЧАТЬ ЧИСЕЛ С НУЛЯМИ СЛЕВА (ПО ПЕРЕМЕННОМУ ФОРМАТУ) **

(*решение на стр.* 85)

Требуется печатать целые положительные и нулевые числа, введенные с перфокарт (колонки с 1 по 9) на 9 позициях с нулями слева (если они необходимы). Воспользуемся для этих целей свойствами переменного формата.

2.7. КОМПЬЮТЕР — ХУДОЖНИК *,***?

(*решение на стр.* 87)

Написать программу, позволяющую напечатать портрет, воспроизведенный на рис. 23.

```
11. 21. 31. 41. 51. 12. 22. 32. 42. 52. 13. 23. 33. 43. 53. 14. 24. 34. 44. 54. 15. 25. 35. 45. 55. 16. 26. 36. 46. 56.

11. 21. 31. 41. 51. 12. 22. 32. 42. 52. 13. 23. 33. 43. 53. 14. 24. 34. 44. 54. 15. 25. 35. 45. 55. 16. 26. 36. 46. 56.

11. 12. 13. 14. 15. 16. 21. 22. 23. 24. 25. 26. 31. 32. 33. 34. 35. 36. 41. 42. 43. 44. 45. 46. 51. 52. 53. 54. 55. 56.

 11.   12.   13.   14.   15.   16.
 21.   22.   23.   24.   25.   26.
 31.   32.   33.   34.   35.   36.
 41.   42.   43.   44.   45.   46.
 51.   52.   53.   54.   55.   56.

T(1,1) =   11.  T(1,2) =   12.  T(1,3) =   13.  T(1,4) =   14.  T(1,5) =   15.  T(1,6) =   16.
(2,1) =   21.  T(2,2) =   22.  T(2,3) =   23.  T(2,4) =   24.  T(2,5) =   25.  T(2,6) =   26.
(3,1) =   31.  T(3,2) =   32.  T(3,3) =   33.  T(3,4) =   34.  T(3,5) =   35.  T(3,6) =   36.
(4,1) -   41.  T(4,2) =   42.  T(4,3) =   43.  T(4,4) =   44.  T(4,5) =   45.  T(4,6) =   46.
(5,1) =   51.  T(5,2) =   52.  T(5,3) =   53.  T(5,4) =   54.  T(5,5) =   55.  T(5,6) =   56.

T(1,1) =   11.  T(1,2) =   12.  T(1,3) =   13.  T(1,4) =   14.  T(1,5) =   15.  T(1,6) =   16.
T(2,1) =   21.  T(2,2) =   22.  T(2,3) =   23.  T(2,4) =   24.  T(2,5) =   25.  T(2,6) =   26.
T(3,1) =   31.  T(3,2) =   32.  T(3,3) =   33.  T(3,4) =   34.  T(3,5) =   35.  T(3,6) =   36.
T(4,1) =   41.  T(4,2) =   42.  T(4,3) =   43.  T(4,4) =   44.  T(4,5) =   45.  T(4,6) =   46.
T(5,1) =   51.  T(5,2) =   52.  T(5,3) =   53.  T(5,4) =   54.  T(5,5) =   55.  T(5,6) =   56.

 11.   21.   31.   41.   51.
 12.   22.   32.   42.   52.
 13.   23.   33.   43.   53.
 14.   24.   34.   44.   54.
 15.   25.   35.   45.   55.
 16.   26.   36.   46.   56.

 11.     22.     33.     44.     55.

 11.
 22.
 33.
 44.
 55.

 11.
        22.
                33.
                        44.
                                55.
```

Рис. 19

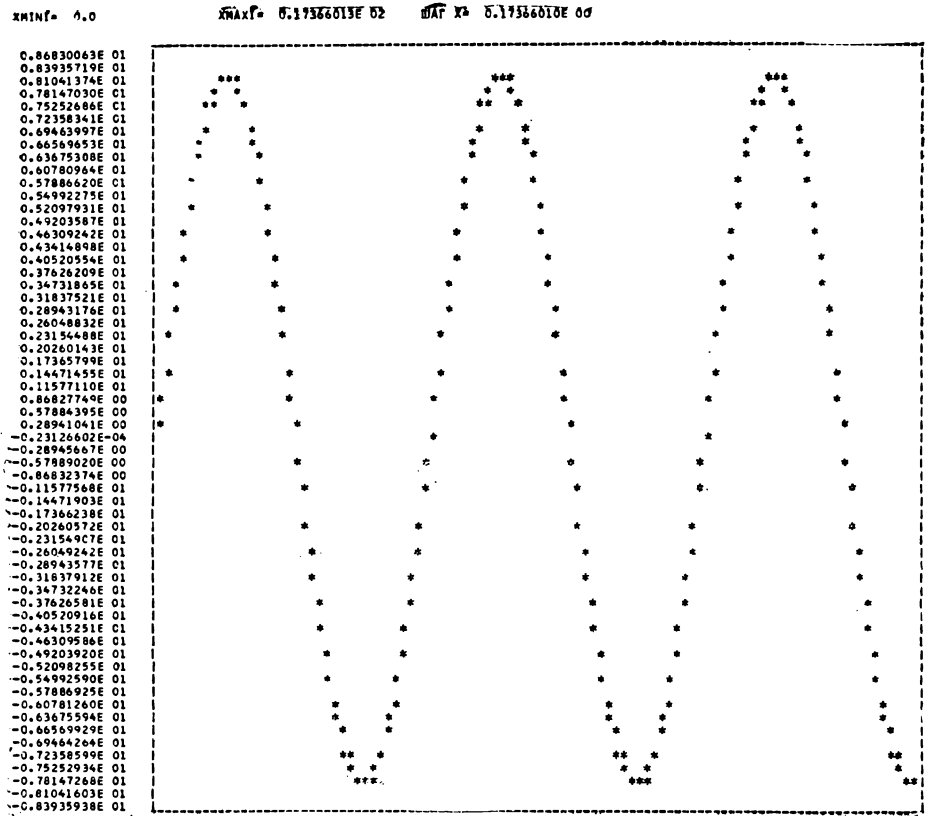

XMINF= 0.0
XMAXF= 0.17366013E 02
DAF X= 0.17366013E 00
0.86830063E 01
0.83935719E 01
0.81041374E 01
0.78147030E 01
0.75252686E 01
0.72358341E 01
0.69463997E 01
0.66569653E 01
0.63675308E 01
0.60780964E 01
0.57886620E 01
0.54992275E 01
0.52097931E 01
0.49203587E 01
0.46309242E 01
0.43414898E 01
0.40520554E 01
0.37626209E 01
0.34731865E 01
0.31837521E 01
0.28943176E 01
0.26048832E 01
0.23154488E 01
0.20260143E 01
0.17365799E 01
0.14471455E 01
0.11577110E 01
0.86827749E 00
0.57884395E 00
0.28941041E 00
-0.23126602E-04
-0.28945667E 00
-0.57889020E 00
-0.86832374E 00
-0.11577568E 01
-0.14471903E 01
-0.17366238E 01
-0.20260572E 01
-0.23154907E 01
-0.26049242E 01
-0.28943577E 01
-0.31837912E 01
-0.34732246E 01
-0.37626581E 01
-0.40520916E 01
-0.43415251E 01
-0.46309586E 01
-0.49203920E 01
-0.52098255E 01
-0.54992590E 01
-0.57886925E 01
-0.60781260E 01
-0.63675594E 01
-0.66569929E 01
-0.69464264E 01
-0.72358599E 01
-0.75252934E 01
-0.78147268E 01
-0.81041603E 01
-0.83935938E 01

Рис. 20

числу, не исключенному ранее (в данном случае 3), и исключим из массива T все числа, кратные ему; точно так будем поступать и далее. Разумеется, бесполезно продолжать этот процесс до T(3000); можно его прекратить после T(M), где M — корень квадратный из наибольшего точного квадрата, ближайшего к 3000 снизу.

Проиллюстрируем этот метод на примере массива, уменьшенного до 30 элементов. Следовательно, необходимо пройти от T(2) до T(5), поскольку 5 — это квадратный корень из 25, самого большого точного квадрата, меньшего 30.

```
исходный массив

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 ... 29 30

после первого просмотра

1 2 3 0 5 0 7 0 9  0 11  0 13  0 15  0 17  0 19  0 21 ... 29  0
  ↑
после второго просмотра

1 2 3 0 5 0 7 0 0  0 11  0 13  0  0  0 17  0 19  0  0 ... 29  0
    ↑
после третьего просмотра

1 2 3 0 5 0 7 0 0  0 11  0 13  0  0  0 17  0 19  0  0 ... 29  0
      ↑
```

И наконец, все ненулевые числа группируются в левой части массива и затем печатаются.

3.4. СОВЕРШЕННЫЕ ЧИСЛА *,**

(*решение на стр.* 98)

Совершенными числами называются числа, равные сумме своих делителей, включая 1. Например, 6 — совершенное число, поскольку $6 = 1 + 2 + 3$. Требуется найти и напечатать вместе с их делителями совершенные числа, лежащие в диапазоне от 4 до 10 000.

Напишите первое решение, не особенно стремясь к оптимизации вычислений. После того как программа пройдет через машину, вы убедитесь, что время вычисления весьма велико. Это заставит вас подумать над оптимизацией и прийти ко второму решению.

3.5. ДИХОТОМИЧЕСКИЙ ПОИСК В ТАБЛИЦЕ **

(*решение на стр.* 104)

Таблица содержит пары значений (X, Y), причем значения X расположены в возрастающем порядке. Требуется написать подпрограмму-функцию TABLE, которая по аргументам:

N: число пар (X, Y)
A: аргумент (ключ) поиска
X: таблица значений X
Y: таблица значений Y
J: индикатор ошибки

дает значение Y, соответствующее ключу A, если последнему найден точный эквивалент в таблице X. Тогда J принимает значение 0; в противном случае результат не определен и J принимает значение 1.

Будем использовать следующий метод:

· определим значение K, такое, что

$$2^K < N \leqslant 2^{K+1},$$

· начнем с $I = N$ и проверим равенство $X_I = A$;
· если равенство выполняется, то Y_I — искомое значение,
· в противном случае вычислим новое значение I:

$$I = I - 2^K \quad \text{если} \quad A < X_I,$$

$$I = I + 2^K \quad \text{если} \quad A > X_I,$$

· до тех пор, пока не выполнится отношение $A = X_I$, будем повторять те же вычисления с 2^{K-1}, затем с 2^{K-2}, ... и, наконец, с 2^0.

В процессе вычислений I может принимать значения, выходящие за границы интервала (1, N). Тогда необходимо скорректировать I, применяя тот же метод до тех пор, пока значение I вновь не окажется в требуемом интервале. В этом случае исключается сравнение A с X_I, поскольку X_I оказывается за пределами массива.

Таким образом, искомое значение будет найдено не более чем за K попыток ($K = 5$ при $N = 100$). Если после $K + 1$ попыток значение X_I, соответствующее A, все еще не найдено, то его не существует.

Следующий пример иллюстрирует механизм поиска. Пусть $Y = 18$ и $X_1 = 1$, $X_2 = 2$, $X_3 = 3$, ..., $X_{18} = 18$, а $A = 1$. Число 18 попадает между 16 и 25, и поэтому $K = 4$.

$I = 18$	$A < X_{18}$,
$I = 18 - 2^4 = 2$	$A > X_2$,
$I = 2 - 2^3 = -6$	I необходимо корректировать,
$I = -6 + 2^2 = -2$	I необходимо корректировать,
$I = -2 + 2^1 = 0$	I необходимо корректировать,
$I = 0 + 2^0 = 1$	$A = X_I$; решение получено.

3.6. ПОИСК НАИБОЛЬШЕГО ЭЛЕМЕНТА В ТРЕХМЕРНОМ МАССИВЕ*

(*решение на стр.* 105)

Пусть массив T имеет размерность (3, 5, 7). Найти наибольшее содержащееся в нем число и напечатать его. Предпочтение следует отдать *быстрому* решению.

3.7. ПОИСК ДВУХ ОДИНАКОВЫХ ЧИСЕЛ В ДВУМЕРНОМ МАССИВЕ*

(*решение на стр.* 106)

Массив A(30, 7) содержит два (и только два) одинаковых числа. Таким образом, требуется напечатать их индексы. Обратите внимание на то, чтобы никакой элемент массива не сравнивался сам с собой!

3.8. СУММИРОВАНИЕ ЭЛЕМЕНТОВ ДВУМЕРНОГО МАССИВА, СУММА ИНДЕКСОВ КОТОРЫХ РАВНА ЗАДАННОЙ КОНСТАНТЕ**

(*решение на стр.* 108)

Массив X(10, 30) содержит вещественные числа (типа REAL). Требуется ввести с перфокарты целое число K и вычислить сумму элементов $X_{i,j}$, для которых $i + j = K$. Прежде, однако, следует убедиться, что значение K позволяет найти решение, в противном случае нужно напечатать сообщение об ошибке. Значения элементов массива X вводятся предварительно и подготовлены на 30 перфокартах следующим образом:

- с одной перфокарты заполняется один столбец массива,
- каждое число занимает 8 колонок на карте.

Разумеется, необходимо найти искусное решение.

3.9. КВАДРАТНЫЙ КОРЕНЬ*

(*решение на стр.* 110)

Вычислить квадратный корень из произвольного вещественного числа, прочитанного с перфокарты, используя метод Ньютона:

$$Y_{n+1} = \frac{1}{2}\left(Y_n + \frac{A}{Y_n}\right)$$

и полагая $Y_0 = A$.

Если A — отрицательное число, то необходимо напечатать сообщение об ошибке.

Результат должен иметь относительную точность $1 \cdot 10^{-6}$. Как только получено значение Y_1, оно используется для получения следующего приближения Y_2, по которому в свою очередь вычисляется Y_3, ... и так до тех пор, пока не будет достигнута требуемая точность.

Замечание. Этот метод всегда сходится.

3.10. ВЫЧИСЛЕНИЕ ЧИСЛА e**

(*решение на стр.* 111)

Определить число e — основание натуральных логарифмов с помощью ряда:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}$$

для всех значений n от 1 до 20. Для каждого случая напечатать n и соответствующее e; повторить вычисления с двойной точностью.

3.11. ПОСЛЕДОВАТЕЛЬНОСТИ БЕЗ ПОВТОРЕНИЙ*

(*решение на стр.* 112)

Подготовить, а затем напечатать одномерный массив, содержащий все числа из 4 цифр, причем внутри одного числа не должно быть двух одинаковых цифр (этому требованию удовлетворяют 5040 чисел). Например, к ним относятся: 0123, 2714, 4902. Но число 24 такой последовательностью не является (оно начинается с *двух* нулей) [1].

3.12. ВЫЧИСЛЕНИЕ ИНТЕГРАЛА МЕТОДОМ ТРАПЕЦИЙ*

(*решение на стр.* 113)

Рассмотрим кривую $Y = f(X)$ (рис. 24).
Требуется вычислить интеграл:

$$S = \int_{X_0}^{X_1} f(X)\, dX,$$

т. е. определить площадь, заключенную между кривой, осью X и линиями, проектирующими точки A и B на эту ось (соответствующий участок на рисунке заштрихован).

---

[1]) Тот, кого заинтересует эта задача, может найти ее в более общей постановке в книге Н. Вирта «Систематическое программирование. Введение», М., «Мир», 1977, § 15.4. — *Прим. ред.*

Если f(X) — достаточно простое математическое выражение и вы сильны в математике, то воспользуйтесь аналитическим методом, и задача решена. В противном же случае придется применить один из многих численных методов и получить приближенное, но достаточно точное решение. Среди

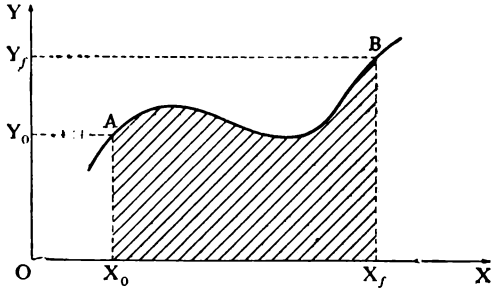


Рис. 24

этих методов одним из самых простых (но и из наименее точных) является метод трапеций. Интервал (X_0, X_f) разбивается на равные подынтервалы ΔX, и рассматриваются элементарные заштрихованные трапеции (см. рис. 25). Сумма их

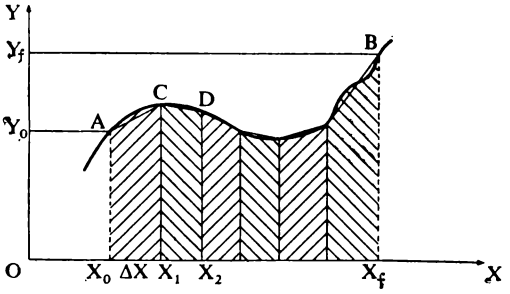


Рис. 25

площадей весьма близка к искомому значению, если интервалы ΔX достаточно малы. Площадь первой трапеции равна:

$$\frac{AX_0 + CX_1}{2} * (X_1 - X_0) = \frac{Y_0 + Y_1}{2} \cdot \Delta X.$$

Сумма элементарных площадей записывается следующим образом:

$$\Delta X \cdot \left(\frac{Y_0 + Y_1}{2} + \frac{Y_1 + Y_2}{2} + \frac{Y_2 + Y_3}{2} + \ldots \right.$$

$$\left. \ldots + \frac{Y_{n-2} + Y_{n-1}}{2} + \frac{Y_{n-1} + Y_n}{2} \right),$$

откуда

$$S = \Delta X \cdot \left(\frac{Y_0}{2} + Y_1 + Y_2 + Y_3 + \ldots + Y_{n-2} + Y_{n-1} + \frac{Y_n}{2}\right).$$

Таким образом, все свелось к вычислению значений $Y = f(X)$ для следующих значений X:

$$X_0,\ X_0 + \Delta X,\ \ldots,\ X_0 + (n-1)\Delta X,\ X_0 + n\Delta X = X_f$$

Нам необходимо написать подпрограмму-функцию (с именем SØMME), которая даст значение S в зависимости от аргументов функции:

X_0: начальная абсцисса,
X_f: конечная абсцисса,
n: число элементарных интервалов,
F: имя подпрограммы типа FUNCTIØN, вычисляющей $Y = f(X)$.

Чтобы подпрограмма SØMME нашла практическое применение, она должна «интегрировать» любую функцию f(X). Сделать это не трудно, поскольку, как нам известно, среди аргументов одной подпрограммы могут быть имена других подпрограмм.

Приведем пример программы, использующей функцию SØMME:

```
      EXTERNAL TRUC
      ..............
      VAL = SØMME (- 3., 17., .01, 20, TRUC)
      PRINT 21, VAL
21    FØRMAT (...
      ..........
      END

      FUNCTIØN TRUC (A)
      ............
      TRUC = ... f(A)
      RETURN
      END
```

3.13. РЕШЕНИЕ НИЖНЕЙ ТРЕУГОЛЬНОЙ СИСТЕМЫ*

(*решение на стр.* 115)

Требуется написать подпрограмму, которая решает треугольную систему порядка N:

$$a_1^1 X_1 = b_1,$$
$$a_1^2 X_1 + a_2^2 X_2 = b_2,$$

$$a_1^3X_1 + a_2^3X_2 + a_3^3X_3 \qquad = b_3,$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$
$$a_1^nX_1 + a_2^nX_2 + \ldots + a_n^nX_n = b_n.$$

Подпрограмма имеет следующие аргументы:
A: квадратная матрица коэффициентов,
B: вектор свободных членов,
X: вектор решения,
N: порядок системы,
K: индикатор ошибки.

Если все кончилось благополучно, то индикатор K будет равен 0. В противном случае он будет иметь значение 1. Это означает, что в какой-то момент один из делителей оказался равным нулю. Алгоритм чрезвычайно прост. Достаточно найти X_1 из первого уравнения, подставить его во второе, чтобы получить X_2 и продолжать таким образом до тех пор, пока не будет вычислено значение X_N.

Глава 4

ВАРИАЦИИ НА ТРИ ТЕМЫ

4.1. ТЕМА ПЕРВАЯ: СОРТИРОВКА

4.1.1. СОРТИРОВКА ПОСРЕДСТВОМ ПЕРЕСТАНОВКИ СОСЕДНИХ ЧИСЕЛ *

(*решение на стр.* 116)

Требуется отсортировать массив, содержащий 1000 чисел, используя следующий метод [1]).

Будем производить последовательные просмотры массива и каждый раз пару за парой сравнивать соседние числа. Если числа в паре расположены в порядке возрастания, оставляем их без изменения; в противном случае меняем их местами. Затем (в любом случае) перейдем к следующей паре. Сортировка считается оконченной, если в ходе просмотра не была произведена ни одна перестановка.

Приведем пример массива из 6 чисел:

```
исходный массив:          1 7 6 4 2 5
                            6 7
                              4 7
                                2 7
                                  5 7

после первого просмотра:  1 6 4 2 5 7
                              4 6
                                2 6
                                  5 6

после второго просмотра:  1 4 2 5 6 7
                              2 4

после третьего просмотра: 1 2 4 5 6 7
```

Сортировка окончена, так как во время четвертого просмотра не было совершено ни одной перестановки.

[1]) Этот метод известен как «метод пузырька»; см., например, Кнут Д., «Искусство программирования для ЭВМ», т. 3, М., «Мир», 1978, п. 5.2.2. — *Прим. ред.*

4.1.2. СОРТИРОВКА ПОСРЕДСТВОМ ПЕРЕСТАНОВКИ ПО ИНДЕКСАМ *

(*решение на стр.* 117)

По-прежнему требуется написать программу для сортировки массива из 1000 чисел, используя, однако, иной алгоритм [1]).

Последовательными сравнениями ищется наименьший элемент массива, причем в процессе поиска запоминается индекс текущего наименьшего числа. После того как просмотрено 1000-е число, можно поменять местами 1-е число с наименьшим, индекс которого в этот момент известен. Затем эта процедура повторяется, начиная со второго, с третьего, ..., ..., и, наконец, с 999-го элемента массива.

В случае массива из 5 чисел получим, например:

исходный массив:	5 2 4 1 3	
индекс наименьшего числа:	4	
массив после перестановки:	1 2 4 5 3 |_____| область для просмотра	
индекс наименьшего числа:	2	
массив после перестановки:	1 2 4 5 3 |___| область для просмотра	здесь делается бессмысленная перестановка
индекс наименьшего числа:	5	
массив после перестановки:	1 2 3 5 4 |__| область для просмотра	
индекс наименьшего числа:	5	
массив после перестановки:	1 2 3 4 5	

Таким образом, сортировка массива завершена.

4.1.3. СОРТИРОВКА ДВУМЕРНОГО МАССИВА В СООТВЕТСТВИИ С ИЗМЕНЯЕМЫМ КРИТЕРИЕМ ***

(*решение на стр.* 120)

Требуется написать подпрограмму с названием TRI2DI, предназначенную для сортировки целых чисел, содержащихся в двумерном массиве. Используем тот же алгоритм, что и

---

[1]) В книге Кнута Д., «Искусство программирования для ЭВМ», т. 3, М., «Мир», 1978, п. 5.2.3 этот метод называется «сортировкой посредством простого выбора». — *Прим. ред.*

```
C                    ПРИМЕР ВЫЗОВА ПОДПРОГРАММЫ
C                    ---------------------------
C
      INTEGER TAB(500,11),AUX(11),ERR
C
C **  ЗАПОЛНЕНИЕ МАССИВА ЗНАЧЕНИЯМИ,ВВЕДЕННЫМИ С КАРТ
C
      DO 1 I=1,500
1     READ(5,100,END=50)(TAB(I,J),J=1,11)
      I=501
50    K=I-1
C
C **  СТОЛБЦЫ      5 6 11  И 2  ПРИ СОРТИРОВКЕ РАССМАТРИВАТЬ
C **  В ЭТОМ ПОРЯДКЕ
C
      AUX(1)=5
      AUX(2)=6
      AUX(3)=11
      AUX(4)=2
C **  ОГРАНИЧИТЬ AUX-5-Й ЭЛЕМЕНТ РАВЕН 0
      AUX(5)=0
C
C **  ЕСЛИ НЕОБХОДИМО СОРТИРОВАТЬ 11 СТОЛБЦОВ,ТОГДА ВЕСЬ МАССИВ ВПЛОТЬ
C **  ДО 11-ГО ЭЛЕМЕНТА ЗАПОЛНЯЕТСЯ НОМЕРАМИ НУЖНЫХ
C **  СТОЛБЦОВ
C **
C **  ВЫЗОВ     TRI2DI
C
      CALL TRI2DI(TAB,500,11,K,11,AUX,ERR)
C
C *   СОРТИРУЕМЫЙ МАССИВ
C *   500 ЧИСЛО СТРОК В МАССИВЕ TAB
C *   11  ЧИСЛО СТОЛБЦОВ В МАССИВЕ TAB
C *   K   ЧИСЛО ДЕЙСТВИТЕЛЬНО СОРТИРУЕМЫХ СТРОК
C *   11  ЧИСЛО ЭЛЕМЕНТОВ В СТРОКЕ(ОБЫЧНО СОВПАДАЕТ С ЧИСЛОМ
C *   СТОЛБЦОВ)
C *   AUX-ВСПОМОГАТЕЛЬНЫЙ МАССИВ
C *   ERR- ПРИЗНАК ОШИБКИ,КОТОРЫЙ БУДЕТ ПРОВЕРЯТЬСЯ ПОСЛЕ ВОЗВРАТА К
C *   ВЫЗЫВАЮЩЕЙ ПРОГРАММЕ
C
100   FORMAT(11A4)
C     ... ......
      END
```

Рис. 26

в предыдущем упражнении, но сортировать необходимо только те столбцы массива, которые указаны в данных. Разумеется, переставлять мы будем строки *целиком*[1]). Входные аргументы подпрограммы таковы:

T: двумерный массив, подлежащий сортировке (типа INTEGER),
число строк в массиве T,
J: число столбцов в массиве T,

---

[1]) В сущности речь идет о сортировке по нескольким ключам. Поэтому критерий соответствует ключу и, следовательно, столбцу матрицы (массива). — *Прим. ред.*

I1: число строк, которые действительно будут подвергнуты сортировке (в случае когда сортируется неполный массив),
J1: число элементов в строке (обычно равно J, но разрешается опустить некоторые элементы),
AUX: вспомогательный массив, содержащий индексы последовательных критериев сортировки, критерий, следующий за последним, имеет индекс нуль,
ERR: признак ошибки (переменные ERR и AUX имеют тип INTEGER).

Признак ошибки примет значение 0, если аргументы допускают правильную сортировку, в противном случае он имеет значение 1. На самом деле можно обнаружить следующие ошибки:

- число строк массива меньше 1,
- число столбцов массива меньше 1,
- число строк, которые надо сортировать, меньше 1 либо больше общего числа строк в массиве,
- число столбцов, которые надо сортировать, меньше 1 либо больше общего числа столбцов массива,
- одно какое-либо из значений массива AUX отрицательно или больше числа сортируемых столбцов,
- *первое* значение массива AUX равно нулю.

Чтобы пояснить возможные применения этой подпрограммы, приведем пример вызывающей программы с многочисленными комментариями (см. рис. 26).

4.1.4. СОРТИРОВКА ПОСЛЕДОВАТЕЛЬНЫМИ СЛИЯНИЯМИ **

(*решение на стр.* 120)

Чтобы произвести сортировку одномерного массива IT, состоящего из N элементов, используем следующий метод:

• Упорядочиваем два первых элемента IT в двух первых ячейках рабочего массива ITT, имеющего ту же длину, что и IT. То же самое проделываем с 3- и 4-м элементами и т. д. до последнего элемента IT. Получится массив ITT, содержащий последовательности пар упорядоченных элементов; затем производится слияние пар в IT и образуются «монотонные последовательности» из 4 элементов. Переходя поочередно от IT к ITT, затем от ITT к IT, будем каждый раз удваивать длину упорядоченных последовательностей.

• Когда упорядоченная последовательность станет той же длины N, что и исходный массив, процесс прекращается. Если последовательность находится в IT, то программа завершена;

в противном случае полученную последовательность надо переслать из ITT в IT без изменения.

Пример сортировки небольшого массива IT, состоящего из 7 элементов:

									Длина упорядоченной последовательности
Исходный массив:	(IT)	7	4	6	2	3	4	1	— 1
конец 1-й фазы:	(ITT)	4	7	2	6	3	4	1	— 2
конец 2-й фазы:	(IT)	2	4	6	7	1	3	4	— 4
конец 3-й фазы:	(ITT)	1	2	3	4	4	6	7	— 7

Сортировка завершена, но необходима пересылка из ITT в IT:

(IT)	1	2	3	4	4	6	7

Замечание. Если N не равно степени 2, рано или поздно появится последовательность, которая будет короче всех остальных.

4.1.5. СОРТИРОВКА С ИСПОЛЬЗОВАНИЕМ СТРУКТУРЫ СПИСКА ***

(*решение на стр.* 124)

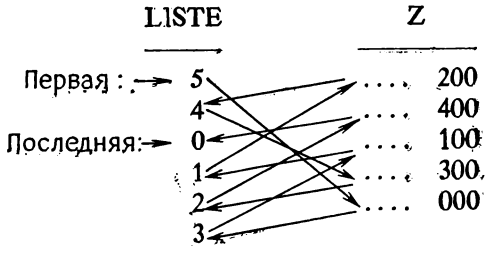
У нас есть колода, содержащая не более 1000 карт, снабженных номерами в интервале от 1 до 99 999 999 в столбцах с 73 по 80. На картах нет двух одинаковых номеров, а некоторые номера вообще отсутствуют. Уронив коробку с картами на пол, мы хотим избежать их сортировки вручную и собираем их как придется, заботясь лишь о том, чтобы они не лежали «вверх ногами». Напишем теперь такую программу сортировки, которая:

- вводит карты;
- размещает их одну за другой последовательно в память в массив Z,
- сортирует их, не перемещая (!?),
- перфорирует их заново в должном порядке.

Для выявления последовательности карт применим метод, подсказанный структурой списка.

Вспомогательный массив LISTE из 1001 слова будет содержать последовательность «адресов» (индексы в массиве) карт с возрастающими номерами. LISTE(1) содержит индекс L, карты с наименьшим номером, LISTE(L + 1) — индекс карты со следующим по величине номером и т. д.; признаком конца цепочки будет значение 0. Рассмотрим метод на при-

мере 5 карт:



LISTE(1) содержит индекс карты с наименьшим номером (здесь Z(*, 5)). Увеличив на 1 этот индекс 5 (что дает 6), получим элемент массива LISTE (т. е. LISTE (6)), в котором находится индекс (3) следующей карты (здесь Z(*, 4)). И так далее до тех пор, пока содержимым LISTE(i) не будет нуль; это означает, что Z(*, i — 1) — последняя карта с наибольшим номером.

Отсюда видно, почему массив LISTE должен содержать 1001 элемент, хотя сортируются только 1000 карт.

4.2. ТЕМА ВТОРАЯ: ВЫЧИСЛЕНИЕ ЧИСЛА π

4.2.1. С ПОМОЩЬЮ МЕДЛЕННО СХОДЯЩЕГОСЯ РЯДА *

(*решение на стр.* 128)

Используется ряд:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots .$$

Провести вычисления сначала с простой, а затем с двойной точностью, ограничиваясь последовательно 100, 1000, 5000, 10 000 и, наконец, 100 000 членами (число членов вводится с перфокарты).

4.2.2. СТАТИСТИЧЕСКИМ МЕТОДОМ *

(*решение на стр.* 130)

Рассмотрим четверть круга единичного радиуса и описанный квадрат. Случайным образом выбираем точки с координатами (X, Y), такими, что $0 \leqslant X \leqslant 1$ и $0 \leqslant Y \leqslant 1$. Если

распределение X и Y равномерно (что предполагается), число K точек, попавших внутрь четверти круга, пропорционально его площади, т. е. $\pi/4$. Отношение $(4 \cdot K)/N$, где N — число случайно выбранных пар точек, дает, таким образом, приближенное значение π.

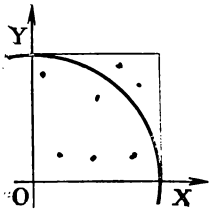


Рис. 27

В качестве датчика случайных чисел используем подпрограмму HASARD, которая при каждом вызове (CALL HASARD(A)) вычисляет новую величину A, заключенную между 0 и 1. Вот листинг этой подпрограммы:

```
      SUBROUTINE HASARD (A)
C     ПОДПРОГРАММА ВЫДАЕТ ОДНО ПСЕВДОСЛУЧАЙНОЕ ЧИСЛО
C
      LOGICAL INIT
      DATA INIT/.TRUE./
      IF (INIT) N=1342773
      INIT=.FALSE.
      N=N*65539+2147483647+1
      A=ABS(N*.4656613E-9)
      RETURN
      END
```

Рис. 28

Замечание. Поскольку здесь речь идет лишь об упражнении, мы нисколько не претендуем на то, что распределение, обеспечиваемое подпрограммой HASARD, на самом деле равномерное. Тем не менее, как это можно констатировать, взглянув на результаты, оно не очень плохое.

Мы сделаем упражнение с 500 000 точками, печатая найденное значение π после каждых 10 000 точек.

4.2.3. МЕТОДОМ ВПИСАННЫХ МНОГОУГОЛЬНИКОВ *

(*решение на стр.* 130)

Рассматривается круг единичного радиуса. Мы знаем, что длина его окружности равна 2π. Будем вписывать в круг многоугольники, увеличивая число их сторон так, чтобы они стремились к совпадению с окружностью (рис. 29).

Известно, что AB равно 2. Вычисляем последовательно:

AC, что дает $\pi_1 = AC \times 2$,

затем AD, — — $\pi_2 = AD \times 4$,

затем AE, — — $\pi_3 = AE \times 8$,

и так далее до тех пор, пока не выполнится неравенство $|\pi_n - \pi_{n-1}| \leqslant 1 \cdot 10^{-8}$.

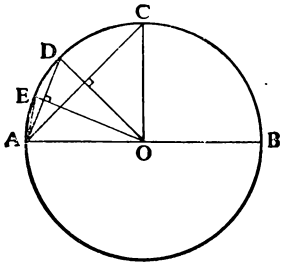


Рис. 29

4.2.4. ПО ФОРМУЛЕ МЕЧИНА *

(*решение на стр.* 133)

Речь идет о формуле, найденной английским математиком Дж. Мечином (1680—1751) (не путать с французским астрономом Мэшеном). Итак, Мечин вывел следующую формулу, которая легко доказывается и быстро дает значения π с очень высокой точностью:

$$\frac{\pi}{4} = 4 \times \operatorname{Arc\,tg} \frac{1}{5} - \operatorname{Arc\,tg} \frac{1}{239}.$$

Для вычисления Arc tg будем использовать не функцию ATAN или ATAN2, имеющуюся в Фортране, а следующее разложение в ряд:

$$\operatorname{Arc\,tg} \frac{1}{x} = \frac{1}{x} - \frac{1}{3x^3} + \frac{1}{5x^5} - \ldots .$$

Напишем программу, используя эту формулу и ограничиваясь 8 членами ряда; начнем с двух членов ряда и напечатаем последовательные значения π при добавлении каждого следующего члена.

4.3. ТЕМА ТРЕТЬЯ: «100 ЗНАЧАЩИХ ЦИФР»

Числа с плавающей запятой имеют обычно от 6 до 9 значащих цифр, числа с фиксированной запятой — от 10 до 12 цифр (18 на CDC 6000). Если же требуется больше значащих цифр, то их можно получить лишь с использованием плавающей запятой с двойной точностью (от 16 до 24). На вычислительных машинах IBM 370 в расширенном Фортране существует даже четырехкратная точность (около 33 цифр).

Если же мы хотим вычислять с 100 значащими цифрами, нужно придумать что-то другое. Отметим сразу, что такое изобилие цифр не представляет слишком большого реального интереса и к тому же время вычисления становится весьма значительным. Но это довольно хорошая иллюстрация того, что можно в случае необходимости сделать при помощи Фортрана. Мы ограничимся здесь случаем целых чисел.

Понадобится несколько слов, чтобы поместить 100 цифр. Чтобы решения, которые мы предложим, годились для большинства машин, мы оставим по 6 цифр на каждое слово. Таким образом, для 100 цифр понадобятся 17 слов, при этом мы получим даже 102 цифры.

4.3.1. ОБРАТНЫЕ ОТ 50 ПЕРВЫХ ЦЕЛЫХ ЧИСЕЛ **

(*решение на стр.* 138)

Требуется вычислить и напечатать обратные целых чисел от 2 до 51. Будем применять следующий метод:

Делим 1 000 000 на N. Частное от деления дает первые 6 цифр результата. Остаток, умноженный на 1 000 000, в свою очередь делится на N, и частное этого деления дает следующие 6 цифр результата. Продолжаем таким образом, пока не получим все требуемые цифры. Например, при N = 19:

$$\frac{1 \times 1\,000\,000}{19} \quad Q = 052\,631 \quad R = 11$$

$$\frac{11 \times 1\,000\,000}{19} \quad Q = 578\,947 \quad R = 7$$

$$\frac{7 \times 1\,000\,000}{19} \quad Q = 368\,421 \quad R = \ldots$$

откуда 1/19 = 0,052631578947368421... .

Напечатаем каждое значение N и соответствующее ему «число» из 102 цифр после 0.

4.3.2. n ФАКТОРИАЛ **

(*решение на стр.* 138)

Вычислим с 102 значащими цифрами n! для n в интервале от 1 до 70. Начнем с того, что поставим 1 в первое слово массива, в остальных словах оставив нули. На этот раз мы действуем с целыми числами, а не с дробными, как в предыдущем упражнении.

Чтобы получить n!, если известно n — 1!, умножаем последовательно каждое слово на n. Если в промежуточном произведении более 6 цифр, то избыточные слева цифры надо при-

бавить к следующему частичному произведению. Таким образом,

```
18 ! = 6 402 373 705 728 000
          728 000 × 19 = 13 832 000
13   + 373 705 × 19 =  7 100 408
 7   +   6 402 × 19 =    0 121 645
```

откуда 19! = 121645100408832000.
Каждый раз печатается n и n!.

4.3.3. ЧИСЛО e, ОСНОВАНИЕ НАТУРАЛЬНЫХ ЛОГАРИФМОВ **

(*решение на стр.* 139)

Вычислить 102 первых десятичных разрядов числа e с помощью ряда, с которым мы уже встречались в упражнении 3.10:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}$$

Лучше соответствующим образом переработать последнее решение упражнения 3.10, стр. 111, чем использовать результаты двух предыдущих упражнений.

Вычисления будут проводиться для n, заключенного между 2 и 70, каждый раз на печать будет выдаваться значение n и соответствующий результат.

Глава 5

ОБРАБОТКА ТЕКСТОВ

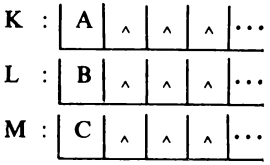
5.1. ОБЩИЕ ПОЛОЖЕНИЯ

В Фортране не признаются, по крайней мере официально, цепочки литер. Таким образом, нельзя декларировать переменную этого типа. Тем не менее существуют «литерные константы» вида nH ..., ' ... ' или * ... *, но их использование чаще всего ограничивается инструкциями FØRMAT, DATA или CALL. Однако существование спецификации A на уровне инструкции FØRMAT позволяет делать многое. К сожалению, из-за различия во внутреннем представлении литер в памяти их обработка весьма серьезно зависит от типа вычислительной машины. Индивидуальность Фортрана проявляется на уровне «слова» памяти, а не на уровне байта. Поскольку, вообще говоря, в языке не предусмотрено никаких средств для доступа к фрагментам информации, меньшим, чем слово, приходится в целях сохранения универсального характера символической программы в слово помещать одну-единственную литеру.

Именно для этого предназначается спецификация A1:

```
      READ 1, K, L, M
1     FØRMAT (3 A1)
```

Если три первые колонки введенной карты содержали литеры «ABC», то в памяти слова K, L, M будут выглядеть так:



Число пробелов справа равно n — 1, где n — число литер в слове используемой машины. Приведем несколько значе-

ний n для некоторых современных вычислительных машин:

CONTROL DATA серии 6000 и 7000: 10

IBM серии 360 и 370: 1, 2, 4, 8

UNIVAC серия 1100: 6

Безусловно, очень жаль, что мы можем использовать лишь малую долю доступного места (1/10 на CDC!). Только в машинах IBM благодаря типу LØGICAL*1 удается не терять место понапрасну; ниже мы рассмотрим это более подробно. Чтобы избавиться от подобной расточительности, можно прибегнуть к подпрограммам обработки литер, написанным либо на машинном языке (ассемблер), либо на Фортране. Рассмотрим случай, когда машинные слова используются полностью:

```
      |  | READ 1, K, L, M
    1 |  | FØRMAT (3A {4 | 6 | 10})   в зависимости от используемой машины
```

Цепочка литер заносится в массив, причем каждая литера попадает в свою позицию. Позиции пронумерованы слева направо, начиная с I:

```
цепочка:                | L A ^ C I G | A L E ^ A Y | A N T ^ C H |
                        +---+---+---+---+---+---+---+---+---+---+---+---+
порядковый номер литеры : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10| 11| 12| 13| 14| 15| 16| 17| 18|
                        +---+---+---+---+---+---+---+---+---+---+---+---+
                                  1               2               3
```

(пример дан применительно к машине UNIVAC 1106).

Достаточно всего лишь двух подпрограмм для осуществления большинства возможных манипуляций с литерами:

- первая отыскивает литеру с номером n и помещает ее в *правую* часть слова, заполняя все позиции слева от нее *пробелами*.
- вторая заменяет n-ю литеру в цепочке литерой из правой позиции некоторого слова, игнорируя все другие позиции слева от нее.

Мы рассмотрим эти подпрограммы в следующем разделе. Теперь же обратимся к случаю ЭВМ IBM 360 и 370. Благодаря декларации LØGICAL*1 можно получить прямой доступ к каждому байту памяти, поскольку машина допускает побайтовую адресацию. Тогда цепочка литер может рассматриваться как массив такого типа. Например:

```
      |  | LØGICAL*1 ZØNE(80)
      |  | READ 1, ZØNE
    1 |  | FØRMAT (80A1)
```

С помощью этих трех инструкций вводится карта, 80 колонок которой помещаются в ZØNE(1), ZØNE(2), ..., ZØNE(80) без какой-либо потери места. Это становится возможным благодаря тому, что в Фортране никогда не проверяется согласованность типа переменной в списке ввода-вывода и соответствующей спецификации в операторе FØRMAT. Таким образом, ничто не мешает оперировать с литерами, используя переменные типа LØGICAL. Однако допускаются лишь сравнения .EQ. или.NE., поскольку применение других операций сравнения будет фиксироваться компилятором как ошибка.

Во внутреннем представлении литеры обычно занимают 6 битов, за исключением машин IBM, в которых они занимают байт, или 8 битов.

5.2. СЕМЕЙСТВО CARCAR **

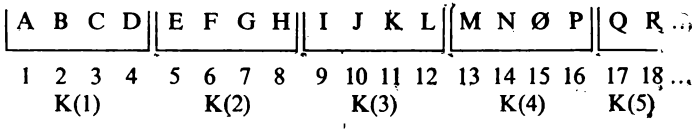
(*решение на стр.* 142)

Мы намерены реализовать упомянутые выше подпрограммы, позволяющие оперировать с отдельными литерами без потери места в машинном слове.

5.2.1. APLCAR

Три аргумента: Z, I, M.

Эта подпрограмма помещает в правую часть слова M I-ю литеру из цепочки Z, при этом левые позиции слова M заполняются пробелами (но не литерами «0» и не двоичными нулями). Если имеется цепочка:



то после вызова подпрограммы

```
| | CALL APLCAR (K, 13, MØT)
```

в словах MØT получим

```
01000000 01000000 01000000 11010100   Двоичный
   ^        ^        ^        M        символьный
```

Наличие пробелов, а не нулей в левой части слова имеет то преимущество, что облегчаются сравнения с литерными константами, определенными в инструкциях DATA:

```
| | DATA IA /4H^^^A/
| | IF (MØT .EQ. IA) GØ TØ ...
```

Не следует писать

```
|  | DATA  A/1HA/
```

либо

```
|  | DATA  A/4HA^^^/,
```

что эквивалентно, поскольку тогда литера оказалась бы *в левой* части слова. Однако некоторые литеры (например, буквы и цифры машины IBM) представляются комбинациями, в которых самый левый бит равен 1, что делает их похожими на отрицательные числа, и это может привести к разным результатам при сравнениях или к переполнению в зависимости от того, с помощью каких машинных команд реализуются компилятором операторы сравнения.

Теперь предстоит написать подпрограмму APLCAR.

5.2.2. SØRCAR

Эта подпрограмма имеет те же самые аргументы, что и APLCAR, но выполняет обратную операцию: помещает в I-ю позицию цепочки Z правую литеру из слова M. Три левые литеры этого слова могут быть какими угодно. Таким образом, если мы вновь воспользуемся массивом K, описанным выше, и напишем инструкции

```
|  | DATA  L/4H^^^Z/
|  | CALL SØRCAR(K, 5, L),
```

то получим цепочку K, измененную следующим образом:

A B C D Z F G H I J K L M N Ø P Q ...

Требуется написать также эту программу SØRCAR.

5.3. PL/1 НА ФОРТРАНЕ

Язык PL/1 был разработан специалистами фирмы IBM. Он оказался чрезвычайно гибким в обработке текстов в основном благодаря таким функциям, как INDEX, SUBSTR, TRANSLATE, ... и т. п.

Если у вас есть вычислительная машина IBM и вы не хотите по той или иной причине пользоваться языком PL/1, ничто не мешает вам написать эти функции на Фортране. Конечно, при этом вы воспользуетесь описанием типа LØGICAL * 1, которое позволяет адресоваться индивидуально к каждому байту.

5.3.1. ФУНКЦИЯ INDEX **

(*решение на стр.* 146)

Написать функцию INDEX, которая

ищет в цепочке	IT
на протяжении	L (литер)
начиная с позиции	N
цепочку литер	ICHCAR
длины	M

и принимает значение, равное номеру позиции в IT первой литеры цепочки ICHCAR, если эта цепочка найдена в IT; в противном случае она принимает значение 0.

Например, после выполнения фрагмента программы:

```
      LØGICAL * 1 Z(21)
      DATA Z/'L', 'A', 'B', 'Ø', 'U', 'R', 'A', 'G', 'E', '^', 'E', 'T',
     1       '^', 'P', 'A', 'T', 'U', 'R', 'A', 'G', 'E'/
      K = INDEX (Z, 15, 7, 'RAGE', 4)
      L = INDEX (Z, 21, 1, 'RAGE', 4)
      M = INDEX (Z, 13, 7, 'RAGE', 4)
      N = INDEX (Z, 17, 5, 'LA', 2)
```

получится: K = 18, L = 6, M = 0, N = 0.

5.3.2. ПОДПРОГРАММА SUBSTR **

(*решение на стр.* 147)

Написать программу SUBSTR, которая

пересылает цепочку литер из массива	IT
начинающуюся с литеры в позиции	N
и имеющую длину	L литер
в массив	ITT
начиная с номера	NN

Пример:

```
      LØGICAL * 1 Z(21), Q(3), W(5)
      DATA Z/'L', 'A', 'B', 'Ø', 'U', 'R', 'A', 'G', 'E', '^', 'E', 'T', '^', '
     1       'P', 'A', 'T', 'U', 'R', 'A', 'G', 'E'/, Q/'E', 'N', 'T'/,
     2       W/'F', 'R', 'E', 'L', 'E'/
      CALL SUBSTR (Q, 1, 3, Z, 1)
      CALL SUBSTR (W, 4, 2, Z, 15)
```

После первого вызова получается

« ENTØURAGE ET PATURAGE »

а после второго вызова:

« ENTØURAGE ET PLEURAGE »

5.3.3 ПОДПРОГРАММА TRANSF **

(*решение на стр.* 147)

Написать подпрограмму TPANSF, которая

в цепочке	IT
начиная с позиции	N
на протяжении	LIT литер

заменяет литеры, которые совпадают с имеющимися в цепочке IDEC, на литеры из соответствующих позиций цепочки IREM, причем две последние цепочки имеют длину L.

Пример программы:

```
      LØGICAL*1 A(28), B(2),C(2)
      DATA A/'L','E','S',' ',' ','P','Ø','U','L','E','T','S',' ',
     1          'R','Ø','T','I','S',' ',' ','S','U','R',' ',' ','C','A',
     2          'N','A','P','E'/,B/'S',' '/,C/' ',' '/
      CALL TRANSF (A, 1, 18, C, B, 2)
```

после выполнения которой получится цепочка:

« LE ^ .PØULET ^ .RØTI ^ .SUR ^ CANAPE »

5.3.4. ИНВЕРТИРОВАНИЕ ЦЕПОЧКИ ***

(*решение на стр.* 148)

Вводится отперфорированная на карте цепочка из 20 литер в массив типа LØGICAL * 1, и надо ее инвертировать так, чтобы первая литера заняла место 20-й, вторая — место 19-й и т. д. В решении используется подпрограмма TRANSF; чтобы помочь читателю, скажем, что задача серьезно усложнилась бы, если бы цепочка была гораздо длиннее.

5.4. МЕРСИ, Г. ДОРНБУШ **

(*решение на стр.* 149)

В замечательной работе, посвященной PL/1 [1]), чтобы проиллюстрировать, как просто на этом языке программируется

[1]) «Le langage PL/1», Dunod, Paris, 1971.

обработка текстов, Маркус Дорнбуш предлагает найти, сколько раз встречается слово MERCI в следующей фразе:

« CAR, POUR REMERCIER LA MERCIERE, TOUS LES MER-
« CIERS DU CANTON SE REUNIRENT. GRANDS MERCIS,
« LEUR DIT LA MERCIERE. L'AMER CITOYEN ENTRA
« DANS LA MERCERIE ET LA REMERCIA AUSSI. NOTRE
« SORT EST BIEN AMER SI PARFOIS DES REMERCIEMENTS
« VOUS RECONFORTENT (...).

Предположив, что эта фраза отперфорирована на трех последовательных картах непрерывным образом, вы вне всяких сомнений покажете, что это также несложно сделать и на Фортране!

5.5. ПОСЛЕДНИЙ СПОСОБ ВЫЧИСЛЕНИЯ ЧИСЛА π**

(*решение на стр.* 150)

Литературная ценность приводимого ниже четверостишья несколько сомнительна, но число букв в каждом из его слов (не учитывая знаки пунктуации) дает первую 31 цифру числа π[1]).

« QUE J'AIME A FAIRE APPRENDRE UN NØMBRE UTILE
AUX SAGES,
« IMMØRTEL ARCHIMEDE, ARTISTE INGENIEUR,
« QUI DE TON JUGEMENT PEUT PRISER LA VALEUR ?
« PØUR MØI, TØN PRØBLEME EUT DE PAREILS AVAN-
TAGES.

Написать программу, которая на любой ЭВМ напечатает сначала четверостишье, а затем первые 31 цифр числа π, считая, что это четверостишье отперфорировано построчно, начиная с 1 колонки на четырех картах.

---

[1]) Перевод этого четверостишья на русский язык не имеет, по-видимому, никакого смысла, тем более, что не удалось бы сохранить главное его достоинство — соответствие между длиной слов и цифрами числа π. Однако приведем два русских текста, которые имеют то же назначение, правда дают меньшее число знаков. Первое: «Это я знаю и помню прекрасно пи многие знаки мне лишни, напрасны», и второе (довольно старое, в нем нужно помнить о букве «ять»): «Кто и шутя и скоро пожелаетъ пи узнать, число ужъ знаетъ». — *Прим. ред.*

5.6. БУКВЕННАЯ ЗАПИСЬ СТА ПЕРВЫХ ЦЕЛЫХ ЧИСЕЛ ***

(*решение на стр.* 151)

Мы хотим напечатать прописью все числа от 1 до 99. Вспомним следующие правила орфографии [1])

- говорят «vingt *et* un» (21), «trente *et* un» (31), ..., «soixante *et* un» (61) и «soixante *et* onze» (71),
- Только «quatre-vingts» (80) пишется с буквой «s» на конце, во всех других случаях: «quatre-vingt-deux» (82), ..., буква s отсутствует.

Названия единиц (1, 2, 3, ...) и десятков (10, 20, ...) будут определены в инструкции DATA. Самое длинное слово для числа единиц имеет 8 букв, самое длинное слово для десятков имеет 12 букв. Можно заметить, что числа от 1 до 99 (в этой программе) можно разбить на три группы (1—19, 20—59, 60—99).

Прежде всего соберем одно за другим подряд все *слова* (не слова памяти, а слова языка), входящие в названия чисел, выраженные буквами, не заботясь об избыточных пробелах, которые могут при этом появиться. Затем уберем эти лишние пробелы так, чтобы каждое *слово* было отделено от следующего за ним всего одним пробелом (для этого можно использовать APLCAR и SØRCAR).

Для записи самого длинного числа (94) требуется 21 буква.

Обратите внимание, что, несмотря на условие задачи, печатается только 99, а не 100 чисел!

5.7. СТАРАТЕЛЬНЫЙ НАБОРЩИК

5.7.1. ПОДГОТОВКА К РАБОТЕ ***

(*решение на стр.* 156)

Строка содержит не более LONG литер. Она начинается *словом* либо одним или несколькими пробелами. Каждое *слово* отделяется от следующего одним пробелом, либо знаком пунктуации, за которым следует один пробел. Последнее слово может быть расположено так, что

- его конец совпадает с концом строки,
- за ним следует один пробел,
- после него стоит точка и, возможно, один или несколько пробелов.

---

[1]) Различие правил записи числительных в русском и французском языках делает невозможным ординарный механический перевод этого упражнения на русский язык. Однако, познакомившись с приемами программирования, читатель может попытаться самостоятельно решить эту задачу применительно к русскому языку. — *Прим. ред.*

Строка помещена в одномерный массив, каждая ячейка (слово) которого содержит максимум литер, допускаемых данной вычислительной машиной.

Надо написать подпрограмму JUSTIF, имеющую три аргумента

```
| | CALL JUSTIF (LIGNE, NCAR, LONG)
```

которая преобразует строку следующим образом:

- если самая крайняя справа отличная от пробела литера — точка, не делается ничего,
- если строка оканчивается одним пробелом, самое правое *слово* сдвигается на одну позицию вправо (таким образом, оно теперь отделено от предыдущего слова слева *двумя* пробелами),
- если строка оканчивается каким-нибудь *словом*, это слово уничтожается вместе с предшествующим ему пробелом.

Проделав это преобразование, надо вставить некоторое число пробелов, равное числу литер уничтоженного слова + 1 (с учетом пробела слева от него). Следует равномерно распределить пробелы между словами, поэтому будем действовать следующим образом:

- попытаемся добавить одинаковое количество пробелов между каждой парой слов.

  Могут представиться три случая:
  - это вполне возможно. Тем лучше!
  - при уничтожении слова получено недостаточно пробелов.
  - после равномерного распределения пробелов между словами остаются лишние пробелы.

В двух последних случаях добавим по одному пробелу между словами, начиная справа. На рис. 30 показаны три возможных случая (см. стр. 61).

Подпрограмма JUSTIF имеет следующие параметры:

LIGNE—строка из N литер (данные и результат);

NCAR —число пробелов, вставленных подпрограммой (результат);

LONG —число литер в строке (данные).

Использование подпрограмм APLCAR и SØRCAR может оказаться полезным.

5.7.2. ВЫПОЛНЕНИЕ РАБОТЫ ***

(*решение на стр.* 158)

Мы хотим напечатать абзац текста из книги, содержащий не более 1500 литер и отперфорированный на последовательных перфокартах так, что не обращается внимание на разры-

```
ELLE VISE A FAVORISER UN CONSTRUCTEUR, EN VULGARISANT UN LANGAGE QUI EST SON OEUVRE. ON DIRA AUSSI QU'ELLE EST MALVENUE
ELLE VISE A FAVORISER UN CONSTRUCTEUR, EN VULGARISANT UN LANGAGE QUI EST SON OEUVRE. ON DIRA AUSSI QU'ELLE EST    9

     CE SERAIT FAIRE BEAUCOUP D'HONNEUR AUX UNIVERSITAIRES QUI PUBLIENT DANS CETTE COLLECTION QUE DE LEUR ATTRIBUER
     CE SERAIT FAIRE BEAUCOUP D'HONNEUR AUX UNIVERSITAIRES QUI PUBLIENT DANS CETTE COLLECTION QUE DE LEUR ATTRIBUER  I

CETTE PHRASE DE 120 CARACTERES SE TERMINE PAR UN MOT DE 25 LETTRES BIEN CONNU: LE VOICI, ICI : ANTICONSTITUTIONNELLEMENT
CETTE PHRASE DE 120 CARACTERES SE TERMINE PAR UN MOT DE 25 LETTRES BIEN CONNU: LE VOICI, ICI :  26
```

Рис. 30

```
     L'INFORMATIQUE EST LA SCIENCE DU TRAITEMENT DE L'INFORMATION. ELLE S'OC
CUPE DE LA MISE DES PROBLEMES SOUS FORME ALGORITHMIQUE, ET DE LEUR EXPLOITATION
SUR ORDINATEURS. LA PROGRAMMATION EST LE LIEN ENTRE LES NOTIONS SEMANTIQUES, PRO
PRES A L'HOMME, ET LES ASPECTS FORMELS OU SYNTAXIQUES, CARACTERISTIQUES NON SEUL
EMENT DES ORDINATEURS, MAIS DE TOUTE L'INFORMATIQUE. C'EST POURQUOI TOUT LANGAGE
 DE PROGRAMMATION JOUE UN DOUBLE ROLE. IL SERT D'ABORD A DIRE DES ALGORITHMES: L
ES PROGRAMMES REDIGES DANS CES LANGAGES DNT POUR NOUS UN SENS. A CE NIVEAU, UN L
ANGAGE DE PROGRAMMATION SERT DE MOYEN DE COMMUNICATION ENTRE HOMMES, ET PEUT ETR
E UTILISE POUR LA PUBLICATION D'ALGORITHMES. IL LE FAIT D'AUTANT MIEUX QUE SA DE
FINITION EST PLUS STRICTE ET SE PRETE MOINS A DES INTERPRETATIONS VARIEES. ALGOL
 A JOUE CE ROLE, ET LE JOUE ENCORE TANT QU'IL RESTE UN DES PLUS PUISSANTS ET DES
 MIEUX DEFINIS DES LANGAGES DE DESCRIPTION DES ALGORITHMES NUMERIQUES.
```

Рис. 31

вы слов (и не добавляется знак переноса). Разумеется, при этом соблюдаются обычные правила пунктуации, в соответствии с которыми все знаки препинания, за исключением апострофа, пишутся сразу вслед за предшествующим словом и отделяются от последующего одним пробелом.

Текст должен быть выравнен, т. е. начала и концы всех строк должны располагаться друг под другом. Таким образом, все строки имеют одно и то же число литер L ($32 \leqslant \leqslant L \leqslant 132$). Значение L вводится с перфокарты (расположенной после карт, содержащих текст) и кратно 4 для машин IBM, 6 для машины UNIVAC и т. д.

Литеры одна за другой переносятся во вспомогательный массив из L литер. Если строка целиком содержит пробелы или если литера, следующая за последней перенесенной литерой, является пробелом, то строка печатается такой как есть. В противном случае нужно вызвать подпрограмму JUSTIF (см. первую часть данного упражнения), чтобы привести строку в соответствие с заданной длиной L и только тогда напечатать ее. Затем программа продолжает обработку текста со слова, исключенного подпрограммой JUSTIF (его длина сообщается во втором аргументе этой подпрограммы).

На рис. 31 приведен пример текста, который можно использовать в качестве исходных данных для этого упражнения; этот текст нанесен на перфокарты, расположенные в том порядке, как показано на рисунке. (Текст заимствован из предисловия, написанного Ж. Арсаком к книге П. Берте: «Язык программирования PL/1», DUNOD, Paris, 1973.)

глава 6

ОБРАБОТКА ФАЙЛОВ

6.1. СЛИЯНИЕ ДВУХ ФАЙЛОВ *

(*решение на стр.* 161)

Два файла F1 и F2 имеют одинаковую структуру и расположены соответственно на логических устройствах 8 и 10. Требуется слить их в один файл F3 и поместить его на логическое устройство 11.

Каждая запись состоит из 8 цифровых литер — поле R1 и 32 буквенных литер. Записи расположены в порядке возрастания их числового ключа R1.

Чтобы осуществить слияние, запись из F1 считывается в область Z1, а запись из F2 — в область Z2. Сравниваются числовые ключи R1 в Z1 и Z2. Предположим, что ключ записи, считанный в Z2, меньше соответствующего ключа в Z1. Тогда запись из Z2 копируется в F3. Затем снова считываем запись из входного файла (в данном случае из F2) и продолжаем по той же схеме, пока не дойдем до конца одного из двух файлов F1 или F2. Начиная с этого момента копируются записи, которые могут еще остаться в другом файле.

Число записей в F1 и F2 неизвестно. Поэтому требуется подсчитать и напечатать число записей, прочитанных из F1 и F2, а также напечатать весь файл F3.

Каждый файл содержит по меньшей мере одну запись.

6.2. СЛИЯНИЕ n ФАЙЛОВ **

(*решение на стр.* 164)

Имеются N файлов одинаковой структуры, каждый из которых отсортирован в порядке возрастания по некоторому ключу. Каждая запись содержит 20 слов, записанных без формата в «двоичной» системе, причем в первом слове находится ключ (числовой код типа INTEGER).

Зная, что N больше 1 и каждый файл содержит по меньшей мере одну запись, попробуем написать общую программу (в соответствии с алгоритмом из предыдущего упражнения), которая позволит слить эти N файлов в один.

Для отладки этой программы на вычислительной машине можно ограничить (что мы и сделали) число файлов NFIC (NFIC $<$ 5) значением, введенным с перфокарты.

6.3. КОРРЕКТИРОВКА ФАЙЛА НА МАГНИТНОЙ ЛЕНТЕ

(*решение на стр.* 167)

Магнитная лента B1 содержит записи следующего вида:

байт 1	код	«1»
байты 2 — 10	номер товара	9 цифр
байты 11 — 30	назначение товара	20 литер
байты 31 — 36	доступное количество	6 цифр
байты 37 — 42	цена единицы товара	XXX.XX

Эти записи отсортированы в порядке возрастания номера товара.

Другая магнитная лента B2 содержит файл «изменения» за неделю и служит для корректировки файла на ленте B1. Содержащиеся на ней записи составлены по той же схеме, что и на ленте B1, и тоже расположены в порядке возрастания номеров товаров; лишь в первом байте есть отличия, он содержит

«1», если этот товар новый и его нет в файле B1,

«2», если этот товар уже есть в файле B1 (тогда байты 37—42 не имеют никакого значения),

«3», если этот товар исключается (байты 11—42 не имеют значения).

Исходя из файлов B1 и B2 создадим новую, «более современную» версию B1 на третьей ленте B3, действуя следующим образом:

- вставим новые товары точно на их места,
- скорректируем записи типа «2», найдя разность между доступным количеством соответствующего товара на B1 (байты 31—36) и проданным количеством товара на B2 (те же байты),
- исключим записи типа «3».

После этого снова читается лента B3 и ее содержимое печатается для проверки.

Замечания.

1. Если записи на лентах B1 и B2 имеют одинаковый номер товара, но код на B2 равен «1», то товар с ленты B2 заменяет товар с ленты B1.

2. Если обнаружена логическая ошибка в кодах при сопоставлении номеров товаров, то печатаются соответствующие записи B1 и B2 и программа останавливается.

Глава 7

ОБО ВСЕМ ПОНЕМНОГУ...

7.1. 1 2 3 4 5 6 7 8 9 = 100 ***

(*решение на стр.* 170)

Вставить между цифрами 1, 2, 3, 4, 5, 6, 7, 8 и 9, записанными именно в таком порядке, знак одной из четырех арифметических операций: +, —, ×, /, так, чтобы результат восьми последовательных операций равнялся 100. Вычисления производятся шаг за шагом слева направо, причем ни одна операция не имеет приоритета:

```
1 × 2 + 3 × 4 × 5 - 6 + 7 + 8 - 9 = 100
  2   + 3
      5   × 4
          20  × 5
              100 - 6
                  94  + 7
                      101 + 8
                          109 - 9 = 100
```

Имеются восемь решений, которые нужно напечатать, используя в качестве образца первую строчку в приведенной выше таблице.

7.2. ПЛАВАЮЩАЯ ФИКСИРОВАННАЯ ЗАПЯТАЯ **

(*решение на стр.* 174)

После вычислений результат обычно известен с некоторой погрешностью, и можно считать, что только первые 4, 5 или 6 цифр являются значащими. Когда приходится иметь дело (а это случается весьма часто) с числами, которые можно записать с явной десятичной точкой (они не слишком велики и не слишком малы, чтобы был необходим формат «E»), используется формат «F». Но при этом возникают два неудобства:

- надо резервировать 2N + 2 позиций (N = числу значащих цифр + знак + точка),

• если число достаточно велико, то печатаются двоичны знаки, не имеющие никакого смысла.

Пример:

```
      |  | PRINT 1, A, B
    1 |  | FØRMAT (1XF12.5)
```

может дать

```
— 12345.67809
      .00023
```

Чтобы получить более удобное и более осмысленное представление, напишем подпрограмму, которая редактирует заданное число A типа REAL и в массив IT из 8 слов, являющийся аргументом подпрограммы засылает цепочку из LL литер, представляющих собой отредактированное значение A. На рис. 32 имеется несколько примеров того, что мы хотим получить.

```
 16.00000000   -0    16.0000
  1.90000000    3    2.
   .12345600    4    .12
 -1.11110000    7   -1.1111
  1.23456000    5    1.23
  -.65432140    6   -.6543
   .01234500    7    .01234
   .00123450   -0    .001234
   .00012340   -0    .000123
   .00001234    2    .000012
   .00000123    9    .000001
   .00000001   88    .000000
   .00000009   77    .000000
-12.34567000    7   -12.346
123.45670000    6    123.5
-12345.99000000 5   *****
123456.90000000 4   ****
999999999.00000000  3   ***
  -0.00000000   -0    .000000
  -0.00000000   -0    .000000
```

Рис. 32

Общий вид отредактированного результата

SXXX.YYY,

где S — знак. Знак + заменяется пробелом. Если XXX отсутствует, то результат будет иметь вид S.YYY, если отсутствует дробная часть YYY, то будет SXXX.000; если длина цепочки LL недостаточна, то производится округление. Если значение A слишком велико (например, число 123456 надо поместить в 5 позиций), то будут напечатаны звездочки (здесь *****). Нужно было бы 8 позиций: « ∧123456». Если LL < 3 или LL > 8, положим LL = 8.

7.3. ЧИСЛА ПО СПИРАЛИ **

(*решение на стр.* 177)

Заполнить квадратную таблицу T(n, n) последовательными целыми числами от 1 до n^2, расположенными по спирали, начиная с левого верхнего угла и продвигаясь по часовой стрелке:

```
 1  2  3  4  5  6
20 21 22 23 24  7
19 32 33 34 25  8
18 31 36 35 26  9
17 30 29 28 27 10
16 15 14 13 12 11
```

(пример для n = 6).

Будем считать, что размер таблицы не превышает 30×30 элементов; n задается на перфокарте, а полученная таблица печатается.

7.4. ЧИСЛА ПО СПИРАЛИ: КОМПЛЕКСНЫЕ ЧИСЛА ВСЕ УПРОЩАЮТ ***

(*решение на стр.* 181)

Мы хотим получить тот же результат, но другим способом. Будем считать, что T — одномерный массив типа CØMPLEX, имеющий n^2 элементов, действительная часть которых соответствует абсциссе точки квадрата, а мнимая — его ординате. Таким образом, при n = 3 можно рассматривать квадрат на рис. 33, где точки A, B, C, D, ... находятся в центре более мелких квадратов со стороной 1, образующих основной квадрат 3×3 (обратите внимание на направление оси Y).

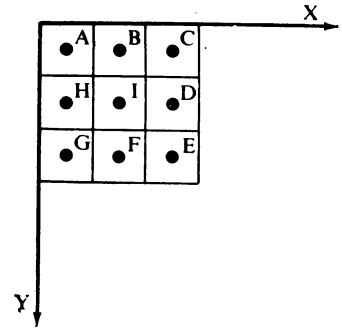


Рис. 33

Координаты этих точек:

A (1, 1)	B (2, 1)	C (3, 1)
H (1, 2)	I (2, 2)	D (3, 2)
G (1, 3)	F (2, 3)	E (3, 3)

Эти пары надо поместить в комплексный массив T из 9 элементов в порядке, соответствующем числам, которые по условию задачи должны были бы попасть в каждый мелкий

квадрат:

(1, 1) (2, 1) (3, 1) (3, 2) (3, 3) (2, 3) (1, 3) (1, 2) (2, 2)
1 2 3 4 5 6 7 8 9

Поскольку мы все время переходим из какой-нибудь ячейки в смежную ячейку, чтобы найти координаты новой точки имея координаты предыдущей, достаточно прибавить к последнему найденному комплексному числу одно из следующих комплексных чисел:

(0, 1) (1, 0) (0, —1) (—1, 0)

в зависимости от того, в каком направлении мы перемещаемся.

7.5. СНАРУЖИ ИЛИ ВНУТРИ? ** [1])

(*решение на стр.* 184)

Рассматривается неправильный многоугольник с 20 сторонами, заданный координатами его последовательных вершин,

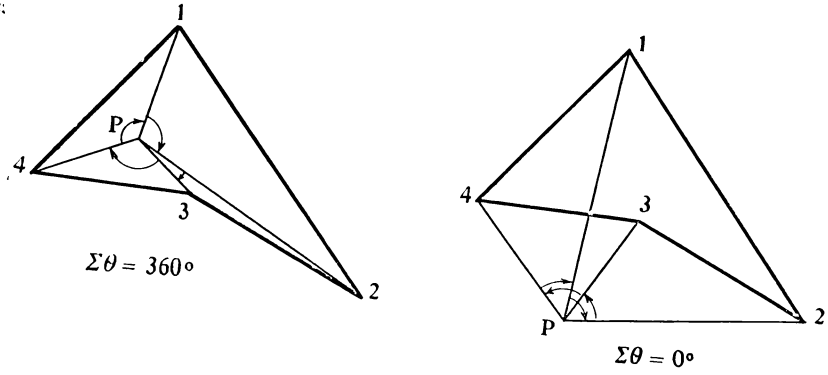


Рис. 34

и некоторая точка P с координатами (X, Y). Не делая чертежа, мы хотим узнать, находится ли точка P внутри или вне многоугольника.

Для этого рассматривается луч из точки P, проходящий по очереди через последовательные вершины многоугольника. Суммируются полученные углы. Если эта сумма равна 0°, точка P находится вне многоугольника, в противном случае сумма, очевидно, равна 360° или —360° (знак зависит от принятого направления обхода) и точка P находится внутри. На

[1]) Идея этой задачи принадлежит Жан-Клоду Рибе из Национального центра научных исследований (CNRS).

рис. 34 показано, что происходит, в (упрощенном) случае многоугольника с 4 сторонами.

Координаты многоугольника вводятся с двух последовательных перфокарт, причем пары (X, Y) перфорируются последовательно, и каждое значение занимает 4 колонки. На третьей карте задаются координаты P. Форма вывода результатов на печать не оговаривается.

7.6. ЮЛИЙ, ГРИГОРИЙ И ИХ КАЛЕНДАРИ ***

(*решение на стр.* 188)

В 45 г. до н. э. декретом Юлия Цезаря было ознаменовано начало хронологической системы, которая названа юлианским календарем. Вследствие расхождения между длительностью календарного года и действительного (тропического) года в 1582 г. календарная дата отставала от истинной на 10 дней. И тогда папа Григорий XIII решил:

- что 15 октября последует за 4 октября (чтобы ликвидировать накопившееся опоздание),
- что отныне периодическое введение високосных годов позволит избежать опоздания [1]).

День принятия григорианского календаря, таким образом, определяется как 1 721 060-й день юлианского календаря.

Зная, помимо того, что високосный год — это такой, у которого число, обозначающее год, делится на 4, за исключением тех, которые кратны 100, но не делятся на 400, написать функцию DATE, которая, исходя из аргументов

IA: год
M: месяц
J: число

даст номер (типа INTEGER) соответствующего юлианского дня. Примем, что IA всегда больше 1583.

Замечание. Эта задача не плод нашей фантазии. Подпрограммы такого типа часто используются астрономами. Она позволяет нам, простым смертным, решать задачи такого рода:

- сколько дней прошло со дня убийства Генриха IV (14 мая 1610 г.) до дня вступления Наполеона в Иену (13 октября 1806 г.)? При помощи нескольких следую-

[1]) На самом деле за 10 000 лет накапливается расхождение в 3 дня. Но для текущих нужд такое расхождение несущественно.

щих инструкций:

```
      INTEGER DATE
      N = DATE (1806, 10, 13) — DATE (1610, 5, 14)
      PRINT 1, N
1     FØRMAT (I10)
      STØP
      END
```

получаем ответ: 71739.

Написав функцию DATE, составить другую программу, использующую ее, которая введет с перфокарты дату (число, месяц, год) в цифровом виде и напечатает соответствующий день недели (понедельник, вторник, ...). Если требуется дата, более ранняя, чем дата введения григорианского календаря, будет напечатано сообщение об ошибке.

7.7. ТРИ ПРОГРАММЫ, ПОЛНЫЕ ОШИБОК, КОТОРЫЕ НАДО НАЙТИ ДО КОМПИЛЯТОРА *,**,***

(*решение на стр.* 192)

Найти в трех программах на рис. 35—37 одни лишь синтаксические ошибки. Можно прибавить себе один балл за

```
          DIMENSION I(3), A(4,5)                          1
          READ (1,1,) I                                   2
10        FORMAT (3HABC , 2X)                             3
          J=ALOG10(23)                                    4
          DO 4567 I=1, J-1, 2                             5
4567      Q=Q+J/(J-2)                                     6
          WRITE (3,10) Q                                  7
          IF (I(1) .EQ. .5) GO TO 3                       8
1         FORMAT (I5)                                     9
          A=I                                            10
          GO TO 47                                       11
13        A = 47                                         12
47        WRITE (3,1) A                                  13
          C=I(3)=178.64                                  14
5         IF (A(3,1)-A(4,6))1, 13, 18                    15
          B=A(2,4)/I(1)                                  16
18        STOP                                           17
          END                                            18
```

Рис. 35

каждую хорошую найденную ошибку и отнять один балл за каждую ложную ошибку. Три программы расположены в порядке возрастания сложности.

Замечание. Инструкции пронумерованы справа, и авторы гарантируют, что номера отперфорированы, начиная с колонки 73.

```
3       FORMAT (1HO, 110//)                                              1
       /DATA L /1., 2., 3., 2*4., 0./                                    2
        INTEGER Z                                                        3
1       DIMENSION X(10,10), B(5), K(10,3), W(30), P(M), L(7)             4
        READ 5 A, W                                                      5
5       FORMAT (E15.8)                                                   6
        N = 0                                                            7
        DO 2, J=1, L(2)                                                  8
        IF (A(J) .EQ. W(J+1) GO TO 3                                     9
        N = N + 1                                                       10
2       CONTINUE                                                        11
        PRINT 3, N                                                      12
        S = 0                                                           13
        READ 5, B                                                       14
10      S = S + B(Z)                                                    15
83      X(2) = B(4)                                                     16
        DO 10 K=1, 5                                                    17
        PRRINT 2, K                                                     18
        N = 0                                                           19
83      M = (N+1)/2                                                     20
        IF (B(M) .EQ. 0.) GOTO 27                                       21
        N=N+1                                                           22
        IF (N.LE.8) GO TO 83                                            23
        L=3                                                             24
        PRINT, 18, B(L-1), B(L+1), B(2*L-1), B(L/2)                     25
18      FORMAT (1X, 2E15,8,F5.0F10.1)                                   26
47      STOP 22                                                         27
        END                                                             28
```

Рис. 36

```
      DIMENSSION (A(30),B(15,C(4.7))                                     1
      COMMON X,Y,Z                                                       2
      LOGICAL TRUC                                                       3
      EQUIVALENCE (A(1),B(4)),(X,B(10)),(Y,A(10))                        4
      DATA TRUC/.FALSE./,I/175./                                         5
      READ (5,6) BI                                                      6
    6 FORMAT(1X,6F6.8)                                                   7
      N=32                                                               8
      DO 3 I=1 10                                                        9
      Z=SQRT(TRUC)-BI*I                                                 10
      IF (Z) 1,2,12                                                     11
      W=ALOG10(N-4)                                                     12
    3 CONTINUE                                                          13
      A(J)-1=B(3)                                                       14
    2 IF(TRUC)IF=5                                                      15
    1 GO TO=27                                                          16
      X1=A(2,4)/W-(1+SIN(B(4)+Z))                                       17
      COSINUS=BI-Z                                                      18
      STOP=READ(10)                                                     19
      RETURN                                                            20
      END                                                               21
```

Рис. 37

Глава 8

НЕСКОЛЬКО НЕЧИСЛОВЫХ ЗАДАЧ

8.1. ЗАДАЧА О ШАРИКЕ И ЗАСЛОНКАХ **

(*решение на стр.* 195)

Шарик бросают в любопытное устройство, схематически представленное на рис. 38. Каждый раз, когда шарик проходит через заслонку, она меняет положение после его прохода.

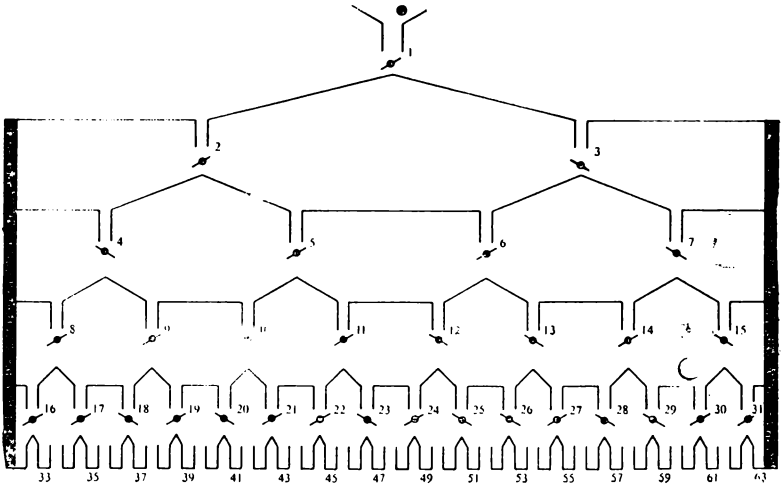


Рис. 38

Когда шарик падает в одну из ячеек с 32 по 63, его подбирают и, не меняя положения заслонок, снова бросают в воронку 1. В предположении, что заслонки расположены, как указано на рис. 38, определить, через сколько «партий» шарик попадет наконец в ячейку 41?

Нужно напечатать число сыгранных партий и расположение заслонок в конечный момент, причем это надо сделать наиболее наглядным способом. Если бы имелось печатающее устройство с литерой «обратная косая черта» (\), это было

бы просто сделать. За неимением такого печатающего устройства можно использовать знак логического отрицания (¬), применяемый в PL/1.

8.2. ВОСЕМЬ ФЕРЗЕЙ НА ШАХМАТНОЙ ДОСКЕ**

(*решение на стр.* 198)

Найти все положения, которые могут занимать восемь ферзей на шахматной доске, когда ни один из них не находится «под боем». На рис. 39 дан пример такого расположения.

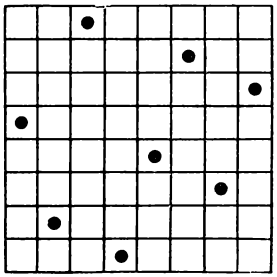


Рис. 39

Нет необходимости представлять шахматную доску двумерным массивом (8,8). Гораздо удобнее разрезать ее на восемь горизонтальных полос, в каждой из которых ферзь может занимать лишь одно положение из восьми возможных. Это положение само может иметь номер от 1 до 8, и мы видим, что достаточно 8 переменных, по одной на полосу, чтобы представить шахматную доску. Таким образом, вычисления проводятся быстрее, чем с одним массивом (8, 8) или с восемью массивами (8).

Имеются 92 решения, которые можно напечатать, выбрав формат, позволяющий наиболее точно воспроизвести действительное расположение на шахматной доске.

8.3. РАЗНОЦВЕТНЫЕ КУБИКИ***

(*решение на стр.* 201)

Один системный программист, возвращаясь с семинара, должен был как-то «убить время», ожидая свой самолет в аэропорту. Он купил в киоске игру, состоящую из четырех кубиков, грани которых раскрашены в четыре разных цвета: белый (blans), зеленый (vert), синий (bleu) и оранжевый (orange). Правила игры такие: надо составить из этих четырех кубиков прямоугольную призму, каждая боковая грань которой раскрашена во все четыре цвета без повторений.

Не обладая большим терпением, но в совершенстве зная возможности вычислительной машины, наш программист быстро отказался от поисков решения, а вернувшись в свой вычислительный центр, он поспешил написать программу, с помощью которой решение было быстро найдено [1]).

[1]) Невыдуманная история!

Расположение кубиков показано на рис. 40. Для простоты мы сделали их развертку.

Попытайтесь решить эту задачу и представить решение примерно в таком же виде, как на нашем рис. 40.

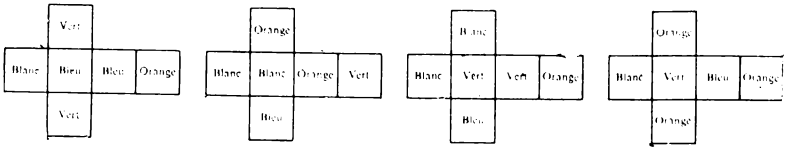


Рис. 40

8.4. ЗАДАЧА О ШАХМАТНОМ КОНЕ (РЕШЕНИЕ В КОМПЛЕКСНЫХ ЧИСЛАХ)***

(*решение на стр.* 206)

В прошлом веке Эйлер предложил общий метод, позволяющий обойти ходом шахматного коня, начиная с произвольно выбранного поля, всю шахматную доску, причем на каждое поле конь попадает всего один раз. Этот метод часто

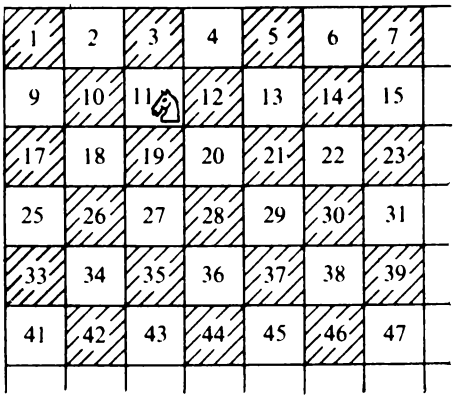


Рис. 41

называют «эвристическим», так как он не всегда приводит к успеху. Он заключается в выборе среди нескольких возможных полей того, откуда имеется минимальное число возможных полей продолжения. Под «полями продолжения» мы понимаем «возможные пути, выходящие из нового испытываемого поля». Рис. 41, на котором представлен фрагмент шахматной доски с перенумерованными клетками, поможет нам уточнить понятие «поля продолжения».

Предположим, что конь находится на поле 11. Затем он может пойти на поля 1, 5, 21, 28, 26 и 17. Перечислим, исходя из этих новых положений, поля продолжения:

из поля	1	имеется	1	поле	продолжения:	18
—	5	—	3	—	—	: 15, 20, 22
—	21	—	7	—	—	: 4, 6, 15, 31, 38, 36, 27
—	28	—	7	—	—	: 13, 22, 38,45, 43, 34, 18
—	26	—	5	—	—	: 9, 20, 36, 43, 41
—	17	—	3	—	—	: 2, 27, 34

Таким образом, надо выбрать поле номер 1.

Требуется найти решение, используя представление каждого положения шахматного коня комплексным числом, действительная и мнимая части которого будут соответственно абсциссой и ординатой поля и, следовательно, будут изменяться от 1 до 8.

Итак, пусть C (типа CØMPLEX) — положение коня в момент t. Чтобы найти его положение в момент t + 1, надо прибавить (по правилам сложения комплексных чисел) к C какое-нибудь из следующих 8 комплексных чисел:

(2, 1) (1, 2) (—1, 2) (—2, 1) (—2, —1) (—1, 2) (1, —2) (2, —1)

За исходное поле примем последовательно каждое из 64 полей шахматной доски и убедимся, что решение существует. В этом случае оно должно быть напечатано наглядным образом с воспроизведением шахматной доски (массив 8×8), в которой каждое поле пронумеровано в порядке прохождения шахматным конем. Если мы зайдем в тупик, будет напечатана лишь пройденная часть массива и сообщение о «катастрофе».

В любом случае затем испытывается следующее исходное поле. Если произошла «катастрофа», ни в коем случае не надо пытаться брать ход назад, чтобы выбрать в некоторый момент другое поле продолжения, так как это повлекло бы за собой весьма сложные вычисления, а опыт показывает, что мы еще не очень в них сильны.

Заметим, наконец, что для удобства полезно выбрать обратное направление оси Y.

Глава 9

ПОЗНАКОМЬТЕСЬ С РЕШЕНИЯМИ...

2. ВВОД-ВЫВОД

2.2. РИСОВАНИЕ КРИВОЙ НА АЦПУ

(*условие задачи на стр.* 24)

Начать следует с поиска максимальных и минимальных значений среди X и Y. С этой целью XMAX и XMIN, а также YMAX и YMIN присвоим значения первых элементов из соответствующих столбцов массива T. Затем в цикле DØ1, используя функции AMAX1 и AMIN1, просмотрим массив T от I = 2 до I = 200. Теперь, чтобы решить вопрос с расположением фигуры на квадратном поле, сравним DX и DY (диапазоны изменения соответственно по X и по Y). Если DX ⩾ DY, то необходимо скорректировать верхний предел по Y [1]). Нижний предел скорректируется автоматически, поскольку вначале для изображения используются наибольшие значения Y. В противном случае нужно скорректировать оба предела по X. (См. программу на рис. 42.)

Одновременно определяется DELTA как наибольшее из значений DX, DY. Затем, имея DELTA и зная вертикальное и горизонтальное расстояния между литерами, можно вычислить шаг по X и шаг по Y. На этом заканчиваются подготовка к печати требуемых величин и оформлению верхней части поля.

Теперь начинается собственно рисование. Нужно напечатать 60 строк по 100 литер в каждой (цикл DØ10I = 1,60). Массив IMP(100) заполняется пробелами; вычисляется YMAX1. В одну строку попадают все те точки, ординаты которых лежат между YMAX и YMAX1. При переходе к каждой следующей строке значение YMAX заменяется на YMAX1.

---

[1]) В программе для этого YMAX умножается на DX/DY. Но во многих случаях такая коррекция приведет к ошибке. В самом деле, если, например, DX = 2, YMIN = 1, YMAX = 2, то после коррекции YMAX = 4, а YMIN = 2, т. е. кривая окажется за пределами поля. Программу следует исправить хотя бы так:

$$YMAX = YMAX + (DX - DY) \cdot 0.5.$$

Те же замечания в полной мере применимы к коррекции XMIN, XMAX. — *Прим. ред.*

В цикле DØ11J выбираются точки для печати и по их абсциссам вычисляются адреса в массиве IMP, куда заносятся звездочки. Одновременно с каждой строкой печатается

```
C *-*     РИСОВАНИЕ КРИВЫХ НА АЦПУ
          DATA IBLAN/1H /,IAST/1H*/
          DIMENSION T(200,2),IMP(100)
          R=3.141592/180.
          DO 50 I=1,200
          T(I,1)=(I-1)*5*R
          T(I,2)=SIN(T(I,1))*8
50        CONTINUE
          XMAX=T(1,1)
          XMIN=XMAX
          YMAX=T(1,2)
          YMIN=YMAX
          DO 1 I=2,200
          XMAX=AMAX1(XMAX,T(I,1))
          XMIN=AMIN1(XMIN,T(I,1))
          YMAX=AMAX1(YMAX,T(I,2))
          YMIN=AMIN1(YMIN,T(I,2))
1         CONTINUE
          DX=XMAX-XMIN
          DY=YMAX-YMIN
          IF(DX.GE.DY) GO TO 15
          DELTA=DY
          XMIN=XMIN*DY/DX
          XMAX=XMAX*DY/DX
          GO TO 16
15        DELTA=DX
          YMAX=YMAX*DX/DY
16        CONTINUE
          PASX=DELTA/100.
          PASY=DELTA/60.
          PRINT 101,XMIN,XMAX,PASX
101       FORMAT(5X,7HXMINI= ,E15.8,5X,7HXMAXI= ,E15.8,5X,7HШАГ X= ,
      1   E15.8,/)
          PRINT 102
102       FORMAT(23X,102(1H_))
          DO 10 I=1,60
          DO 12 J=1,100
12        IMP(J)=IBLAN
          YMAX1=YMAX-PASY
          DO 11 J=1,200
          IF(T(J,2).LE.YMAX1.OR.T(J,2).GT.YMAX) GO TO 11
          K=(T(J,1)-XMIN)/PASX+1
          IF(K.GT.100.OR.K.LT.1) GO TO 11
          IMP(K)=IAST
11        CONTINUE
          PRINT 100,YMAX,IMP
100       FORMAT(E20.8,3X,1H|,100A1,1H|)
10        YMAX=YMAX1
          PRINT 103
103       FORMAT(1H+,22X,102(1H_))
          STOP
          END
```

Рис. 42

соответствующее значение по оси Y и постепенно строятся боковые стороны рамки, окружающей поле.

После выхода из цикла DØ10 остается напечатать нижнюю границу рамки. Из чисто эстетических соображений

используется литера «_» (подчеркивание), и поэтому приходится с помощью литеры «+» отменить переход к следующей строке, чтобы нижняя граница рамки оказалась там, где ей положено быть. Заметим, что это не противоречит условию задачи, поскольку печать кривой уже закончена!

Замечания по поводу оптимизации.

1. Можно не повторять (60 раз) просмотр массива T, если его предварительно отсортировать по Y, расположив ординаты в порядке уменьшения.

2. Конечно, применение функций AMAX1 и AMIN1 привносит элемент изящества, но ради эффективности программу следует изменить так, как предлагается на рис. 43

```
      IF (T(I,1) .GT. XMAX) GO TO 2
      IF (T(I,1) .LT. XMIN) XMIN=T(I,1)
      GO TO 3
2     XMAX=T(I,1)
3     IF (T(I,2) .GT. YMAX) GO TO 4
      IF (T(I,2) .LT. YMIN) YMIN=T(I,2)
      GO TO 1
4     YMAX=T(I,2)
1     CONTINUE
```

Рис. 43

2.3. ПЕРЕПИСЬ С КАРТ НА ЛЕНТУ

(*условие задачи на стр.* 31)

Чтобы не приходилось принимать во внимание количество литер в слове конкретной машины, будем использовать FORMAT (80A1), специфицируя по одной литере в слове. Этот же формат используется и при записи, поскольку желательно, чтобы образ каждой карты на ленте точно соответствовал оригиналу (см. рис. 44).

Когда введена последняя карта, в программе делается переход на метку 2 и на ленту записывается «конец файла». На самом деле не известно, использовалась ли эта лента прежде, и поэтому нужно исключить ситуацию, когда вслед за образами карт могут читаться старые остатки. После перемотки лента читается точно так же, как читалась бы колода перфокарт.

При чтении (с ленты и с перфокарт) используется один и тот же формат. Но при печати формат должен быть иным, поскольку первая литера управляет переходом к следующей строке. Как только обнаруживается признак конца файла, записанный ранее, программа останавливается. Благодаря тому, что чтение и запись ведутся по одной литере (специфика-

ция A1), заметно упростился анализ литер в колонках 23 и 70.

```
C   ПЕРЕПИСЬ С КАРТ НА ЛЕНТУ.
        DIMENSION CARTE(80)
        DATA Z/1HZ/,X/1HX/
3       READ(5,1,END=2) CARTE
1       FORMAT(80A1)
        WRITE(3,1) CARTE
        GO TO 3
2       ENDFILE 3
        REWIND 3
6       READ(3,1,END=4) CARTE
        IF(CARTE(23).EQ.Z.OR.CARTE(70).EQ.X) PRINT 5,CARTE
        GO TO 6
4       REWIND 3
        STOP
5       FORMAT(1X80A1)
        END
```

Рис. 44

Замечание. В нашей задаче каждая *логическая* запись содержит 80 литер, но из этого факта нельзя сделать никаких выводов о длине *физической* записи. В частности, в операционной системе OS IBM программист может по своему усмотрению задать на управляющей карте (DD), относящейся к файлу на ленте, любой *коэффициент блокирования.* Но тогда подпрограммы ввода-вывода, реализующие так называемый «метод доступа», должны сохранить программисту простоту и прозрачность операции. Следующая DD-карта позволяет задать коэффициент блокирования, равный 10:

```
//GØ.FT03F001 DD UNIT = TAPE9, VØL = SER = ME0017,
// DISP = (, KEEP),
// DCB = (RECFM = FB, LRECL = 80, BLKSIZE = 800),
// LABEL = (, SL)
```

2.4. ПЕРЕПИСЬ КАРТ НА ЛЕНТУ С ДОПОЛНИТЕЛЬНОЙ ОБРАБОТКОЙ

(*условие задачи на стр.* 31)

Как и в предыдущей задаче, мы могли бы помещать одну литеру в слово. Но на этот раз у нас будет по 4 литеры в слове, что, впрочем, приемлемо для большинства машин, по крайней мере для достаточно мощных моделей.

Чтобы не усложнять себе жизнь, а у нас имеется 1000 карт, или 200 групп по 5 карт, или, наконец, 800 записей, мы будем вводить сразу по 5 карт. Теперь можно записать решение (рис. 45).

```
C     карты на  ленту с дополн.обработкой.
      DIMENSION CARTE(100)
      DATA DEUX/1H2/
4     READ(5,1,END=2) CARTE
1     FORMAT(20A4)
      WRITE(8,3) CARTE
3     FORMAT(25A4)
      PRINT 8,CARTE
      GO TO 4
2     ENDFILE 8
      REWIND 8
7     READ(8,5,END=6) A
5     FORMAT(97X,A1)
      IF(A.NE.DEUX) GO TO 7
      BACKSPACE 8
      BACKSPACE 8
      READ(8,3) (CARTE(I),I=1,25)
      PRINT 8 ,(CARTE(I),I=1,25)
      READ(8,1)
      GO TO 7
6     REWIND 8
      STOP
8     FORMAT(1X,25A4)
      END
```

Рис. 45

В той части программы, где карты записываются на ленту, заслуживают обсуждения используемые форматы. При чтении применяется принцип, который гласит: «формат должен описывать карту». Если нужно ввести 100 слов (по 20 с карты), то годится спецификация 20A4. Но в таком виде она слишком коротка и поэтому повторится 5 раз, вызывая при каждом повторении ввод новой карты. Когда информация в памяти, она записывается на ленту с иным разбиением, но с применением прежнего принципа. Спецификация (25A4) опять-таки слишком коротка, она повторится 4 раза и вызовет появление четырех записей.

При повторном чтении с ленты мы могли бы использовать часть массива CARTE:

CARTE (с 1 по 24):	колонки с 1 по 96:	24A4
CARTE (25)	колонка 97	A1
CARTE (26)	колонка 98	A1
CARTE (27)	колонки 99—100	A2

Но, по существу, нужна только колонка 98, и поэтому можно использовать спецификацию (97X, A1), чтобы прочитать одну переменную A. В этом случае мы получаем выигрыш во времени за счет пересылки между буфером чтения и массивом в нашей программе.

Чтение ленты продолжается до тех пор, пока не встретится 2 в колонке 98. Когда такая ситуация обнаружена, нужно отступить назад, пропустив две записи: ту, которая только что прочитана и прочитанную перед ней. И вот теперь, поскольку нужно прочитать всю информацию, мы можем с полным правом использовать часть (25 слов) массива CARTE. Нам вполне подойдет формат с меткой 3. Действительно, нет ни малейшего смысла писать еще один формат, если это не печать!

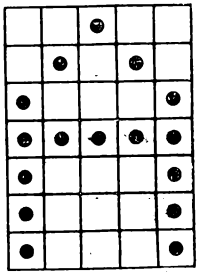


Рис. 46

Рационализаторы упрекнут нас за излишнюю простоту этого решения и отметят, в частности, что

- вначале следовало бы вводить карты по одной и, искусно манипулируя указателями для последовательных карт, обойтись двумя массивами для чтения карт,
- читая ленту попеременно в два массива, можно было бы исключить ее возврат на две записи и реализовать изящную программу «с мигалкой».

Все это так! Но ведь наше упражнение отмечено одной звездочкой! Мы, однако, не будем возражать, если читатель — поборник искусства в программировании — напишет более изысканное решение.

2.5. ДЛЯ ПЕЧАТИ ЗАГОЛОВКА...

(*условие задачи на стр.* 32)

Вначале нужно найти простой и экономный в отношении памяти способ задания «рисунка» каждой литеры. Рассмотрим, например, букву «А», представленную так, как показано на рис. 46. Возникает идея — закодировать начертание каждой буквы с помощью таблички (5×7), элементы которой равны 1 или 0 в зависимости от того, есть что-нибудь в соответствующей клетке или нет. В этом случае из совокупности табличек образуется массив размером 5×7×26=910 слов, но это слишком много.

Чтобы уменьшить размер массива, можно подумать о том, как вместо 35 использовать только 7 элементов на букву. Будем рассматривать каждый «слой» буквы, как десятичное число, в котором 1 соответствует точке (или звездочке), а 0 — пустой клетке. Тогда для описания буквы А потребуются

следующие 7 чисел:

```
0 0 1 0 0
0 1 0 1 0
1 0 0 0 1
1 1 1 1 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
```

Теперь нужно только $7 \times 26 = 182$ слова; но можно добиться еще большего, повернув фигуру на 90°:

```
1 1 1 1 1 0 0
0 0 0 1 0 1 0
0 0 0 1 0 0 1
0 0 0 1 0 1 0
1 1 1 1 1 0 0
```

(если повернуть книгу так, чтобы левый край страницы оказался внизу, и затем посмотреть на эту фигуру, то в ней можно узнать букву А).

На этот раз достаточно $5 \times 26 = 130$ слов. Можно было бы еще более сократить размер массива, отводя для кодирования каждой точки не десятичный, а двоичный разряд. Но, поскольку в слово машины IBM типа 360/370 нельзя поместить число, большее $2^{31} - 1$, такое решение утратило бы свою универсальность [1]).

Чтобы получить доступ к «дескриптору», нужно обратиться в таблицу по номеру буквы, который находится с помощью еще одной дополнительной таблицы (см. программу на рис. 47). Если буква не найдена, то это пробел, либо какая-то литера, не предусмотренная условием задачи и поэтому заменяемая пробелом. Блок-схема программы приводится на рис. 48.

После того как в цикле DØ3 строка печати LIGNE заполнена пробелами, ведется подготовка к занесению звездочек в те места этого массива, которые отмечены в матрице LET, содержащей дескрипторы букв (для каждой буквы из массива NØM). Если буквы в таблице нет (пробел или другая литера), то соответствующее место остается неизменным, что в точности отвечает нашему требованию. Когда в цикле DØ4

[1]) В табличке-дескрипторе 35 клеток. Если каждая клетка кодируется одним двоичным разрядом, то необходимо $5 \times 7 = 35$ разрядов. В слове машины IBM 360/370 только 32 разряда. — *Прим. ред.*

```
      SUBROUTINE TITRE (NOM)
      DIMENSION LETTRE(26),LET(5,26),NOM(12),LIGNE(120)
      DATA LETTRE /1HA,1HB,1HO,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     11HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      DATA BLANC /1H /,LASTER /1H*/
      DATA LET /1111100,1010,1001,1010,1111100,1111111,1001001,1001001,
     11001001,111110,11111C,1000001,1000001,1000001,100010,1111111,
     21000001,1000001,1000001,111110,1111111,1001001,1001001,1001001,
     31000001,1111111,1001,1001,1001,1,111110,1001001,1001001,1001001,
     4110010,1111111,1000,1000,1000,1111111,0,0,1111111,0,0,1000000,
     51000000,1000000,1000000,111111,1111111,1000,10100,100010,
     61000001,111111,1000000,1000000,1000000,1000000,1111111,110,
     71000,110,1111111,1111111,10,100,1000,1111111,111110,1000001,
     81000001,1000001,111110,1111111,1001,1001,1001,110,111110,1000001,
     91010001,1100001,1111110,1111111,1001,11001,101001,1000110,100110,
     11001001,1001001,1001001,110010,1,1,1111111,1,1,111111,1000000,
     21000000,1000000,111111,11111,100000,1000000,100000,11111,111111,
     31000000,110000,1000000,111111,1100011,10100,1000,10100,1100011,
     4111,1000,1110000,1000,111,1100001,1010001,1001001,1000101,
     51000011/
C                                  ПЕЧАТАЕТСЯ        7 СТРОК
      DO 6 K=1,7
C                                  ЗАПОЛНЕНИЕ СТРОКИ ПРОБЕЛАМИ
      DO 3 J=1,120
3     LIGNE(J)=BLANC
C                                  ДВЕНАДЦАТЬ БУКВ В ОДНОЙ СТРОКЕ
      DO 7 I=1,12
C                                  ОПРЕДЕЛЕНИЕ НОМЕРА БУКВЫ
      DO 4 J=1,26
      IF (NOM(I).EQ.LETTRE(J))GO TO 5
4     CONTINUE
      GO TO 7
C                                  КАЖДАЯ БУКВА НА ПЯТИ ПОЗИЦИЯХ
5     DO 2 L=1,5
      M= LET(L,J)/10**(K-1)
      M= MOD(M,10)
      IF (M.EQ.0) GO TO 2
      LIGNE(7*I+L)=LASTER
2     CONTINUE
7     CONTINUE
6     PRINT 9,LIGNE
9     FORMAT (1X, 120A1)
      RETURN
      END
```

Рис. 47

определился номер печатаемой буквы, нужно в массиве LET найти десятичный разряд, соответствующий слою (имеется 7 последовательных горизонтальных слоев), который в данный момент обрабатывается.

В цикле DØ2, который выполняется 5 раз (в каждой букве 5 вертикальных слоев), в младший разряд переменной M помещается цифра 1 или 0 из массива LET. Затем все другие возможные цифры слева от нее исключаются с помощью функции MØD. Если остается 0, то ничего не делается (в строке сохранится пробел). Если же M равно 1, нужно занести звездочку в массив LIGNE в то место, которое соответствует I-й букве в массиве NØM.

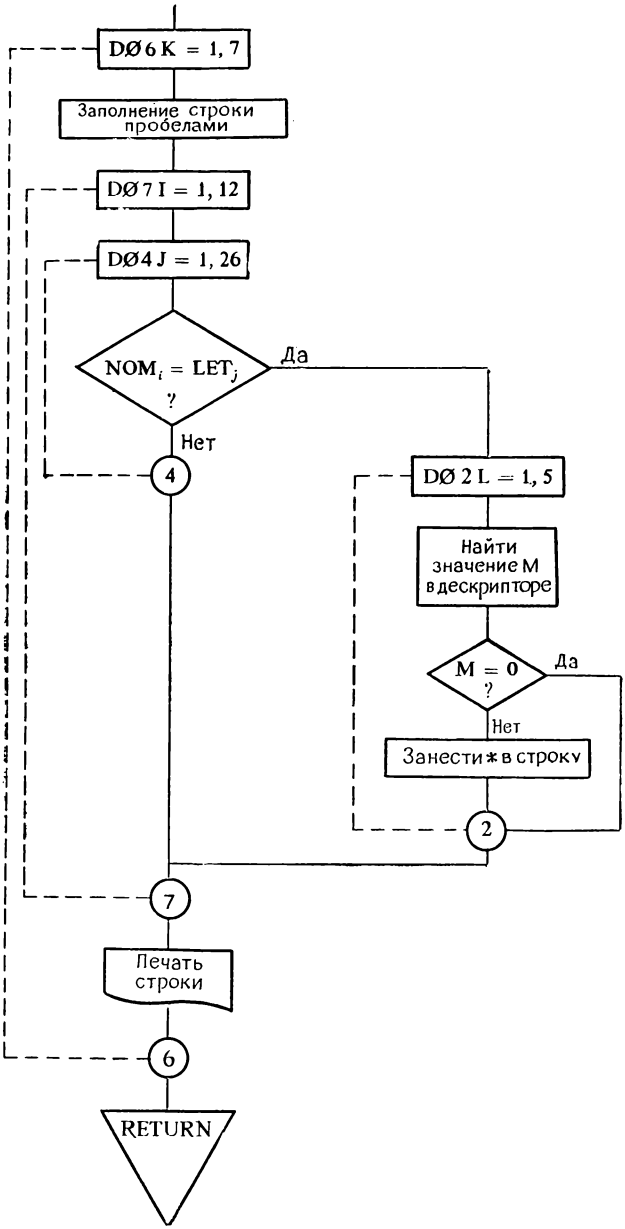

DØ 6 K = 1, 7
Заполнение строки пробелами
DØ 7 I = 1, 12
DØ 4 J = 1, 26
NOM$_i$ = LET$_j$?
Да
Нет
4
DØ 2 L = 1, 5
Найти значение M в дескрипторе
M = 0 ?
Да
Нет
Занести * в строку
2
7
Печать строки
6
RETURN

Рис. 48

Умножая на 7, мы учитываем два пробела, которые разделяют две последовательные буквы на бумаге.

2.6. НУЛИ СЛЕВА (С ПОМОЩЬЮ ПЕРЕМЕННОГО ФОРМАТА)

(*условие задачи на стр.* 32)

Если число M не равно нулю, то формат печати будет построен в соответствии со следующей моделью:

(1Xn (1H0), Im),

где m — количество значащих цифр в M, а n = 9 — m — количество нулей слева.

Если число M равно нулю, будет использоваться заранее заготовленный формат:

(10H^000000000).

Дело в том, что нулевое значение n вообще не было бы воспринято во время выполнения программы или привело бы к неправильным результатам.

Во всех распространенных вычислительных машинах в слове помещается как минимум 4 литеры (за исключением некоторых «мини-компьютеров», в слове которых 16 бит). Поэтому на 4 литеры в слове мы и будем ориентироваться при разбиении переменного формата, который мы составим таким образом, чтобы облегчить доступ к параметрам m и n:

$$\underbrace{(1X,}_{1}\ \underbrace{\wedge\wedge\wedge\wedge}_{2}\ \underbrace{(1H0}_{3}\ \underbrace{),I\wedge}_{4},\ \underbrace{\wedge\wedge\wedge\wedge}_{5}\ \underbrace{)\wedge\wedge\wedge}_{6}$$

Массив из 6 элементов, определенный в инструкции DATA и содержащий «скелет» формата, назовем IMPR. Кроме того, другой массив типа INTEGER из 9 элементов назовем CHIF. В нем также с помощью инструкции DATA зададим в виде литер девять цифр в таком порядке: 1, 2, 3, ..., 9.

В IMPR(5) будем заносить m — количество значащих цифр в числе M, увеличенное на 1, а в IMPR(6) — величину n, которая равна 10 — m (поскольку в Фортране наименьший индекс равен 1). Важно, чтобы эти значения были представлены в виде *литер*, а не в двоичной форме. Программа приводится на рис. 49 (см. стр. 86).

ВОПРОСЫ:

1. Почему написано ALØG10(FLØAT(M)), хотя мы в первой главе условились, что в том Фортране, которым мы

```
C       НУЛИ СЛЕВА С ПОМОЩЬЮ ПЕРЕМЕННОГО  ФОРМАТА
C
        INTEGER IMPR(6), CHIF(9)
        DATA CHIF /1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/,
     1        IMPR/4H(1X,, 4H    , 4H(1H0, 4H),I , 4H    , 4H)
3       READ 1, M
        IF (M.EQ.0) GO TO 2
1       FORMAT (I9)
        I=ALOG10(FLOAT(M))+1
        IF (I.EQ.9) GO TO 6
        IMPR(5)=CHIF(I)
        IMPR(2)=CHIF(9-I)
        PRINT IMPR, M
        GO TO 3
2       PRINT 4
4       FORMAT (10H 000000000)
        GO TO 3
6       PRINT 7, M
7       FORMAT (1X,I9)
        GO TO 3
        END
```

```
 000012345
 000001234
 000000123
 000000012
 000000001
 123456789
 012345678
 001234567
 000123456
 000000000
 400000000
```

Рис. 49

пользуемся, допускается смешение типов в выражениях?
2. Почему отдельно обрабатывается случай, когда в числе M в точности 9 значащих цифр?

ОТВЕТЫ:

1. Потому что математические функции, такие, как ALOG, SIN, SQRT, ..., требуют тем не менее, чтобы их аргументом всегда было значение типа REAL. Аргументы, находящиеся в списке вызова функции или подпрограммы, не подвергаются никаким неявным преобразованиям.
2. Потому что если в формате встретится спецификация вида 0(1H0), результат будет неправильным. Нуль вообще не допускается в качестве коэффициента повторения. (Таким образом, причина та же, что и при M, равном нулю, но в другой спецификации формата.)

2.7. КОМПЬЮТЕР — ХУДОЖНИК

(*условие задачи на стр.* 32)

Чудес нет, и искусство нельзя вложить в рамки уравнений! В одном решении (рис. 50) тупо печатаются искусно подготовленные перфокарты. В другом решении используются с завидным терпением составленные форматы, а программа содержит довольно большое количество инструкций PRINT.

```
      DIMENSION CARTE(20)
    1 READ(5,100,END=500) CARTE
      PRINT 101,CARTE
  100 FORMAT(20A4)
  101 FORMAT(1X,20A4)
      GOTO1
  500 STOP
      END
```

Рис. 50

В этом решении можно с помощью литеры «+» реализовать наложение строк и за счет наложения получить многочисленные оттенки и полутона.

3. НЕСКОЛЬКО КЛАССИЧЕСКИХ ЗАДАЧ

3.1. ПРОСТЫЕ ЧИСЛА: САМОЕ ПРОСТОЕ РЕШЕНИЕ

(*условие задачи на стр.* 34)

Исследуемое число формируется в ячейке IT(I + 1), и поэтому конечное значение цикла DØ1I никогда не должно превышать 499. Далее мы увидим, почему нельзя сразу же написать IT(I + 1) = IT(I) + 2.

Когда вычислено значение IT(I + 1), оно делится на все ранее полученные значения, за исключением 1 и 2. Функция MØD дает нулевой результат, если IT(J) ($3 \leqslant J \leqslant I$) делит IT(I + 1). Тогда к IT(I + 1) нужно прибавить 2. Именно поэтому новое значение IT(I + 1) присваивается двумя инструкциями выше (рис. 52).

Если ни одно из делений не дало нулевого остатка, то произойдет нормальный выход из цикла DØ1J, значение I увеличится на 1 и будет вычислен следующий элемент массива. Если значение I достигло 499, то это означает, что получен 500-й (последнее из искомых простых чисел) элемент массива и этот массив можно напечатать.

ВОПРОС:

Всегда ли допускается выход из внутреннего DO-цикла с входом во внешний цикл, как в данном случае в инструкции IF(...)GØTØ3?

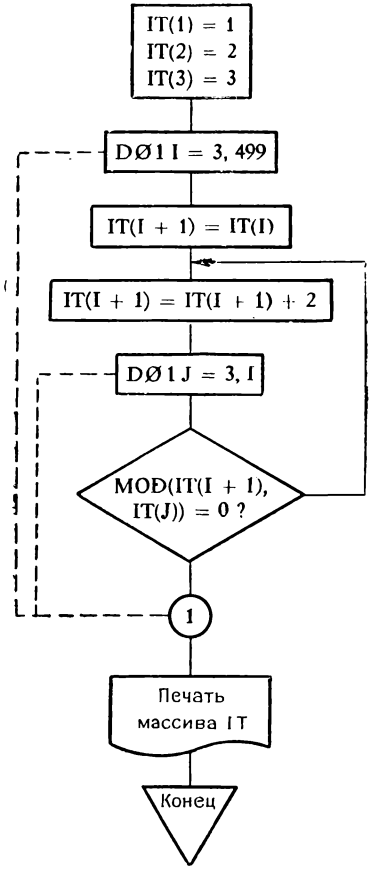


Рис. 51

```
      IT(1)=1
      IT(2)=2
      IT(3)=3
      DO 1 I=3,499
      IT(I+1)=IT(I)
    3 IT(I+1)=IT(I+1)+2
      DO 1 J=3,I
      IF(MOD(IT(I+1),IT(J)).EQ.0) GO TO 3
    1 CONTINUE
      PRINT 50, IT
   50 FORMAT(15I6)
      STOP
      END
```

Рис. 52

ОТВЕТ:

Да, если, как в данном случае, не произошел выход из внешнего цикла.

3.2. ПРОСТЫЕ ЧИСЛА: ПОИСК ЭФФЕКТИВНОГО РЕШЕНИЯ

(*условие задачи на стр.* 34)

Поскольку требуется эффективность, не следует использовать функцию SQRT, лучше каждый раз вычислять NCAR, возводя в квадрат последнее найденное простое число. Вначале значение NCAR равно $3^2 = 9$, а JJ указывает положение элемента, предшествующего числу, квадрат которого находится в NCAR.

NØMB соответствует числу, которое в данный момент исследуется. Если NØMB = NCAR, то очевидно, что оно не простое. В этом случае нужно изменить NCAR, поскольку иначе все последующие простые числа оказались бы больше, чем $\sqrt{\text{NCAR}}$. Чтобы получить новое значение NCAR, нужно взять (JJ + 2)-й элемент массива IT, т. е. написать

```
JJ = JJ + 1
NCAR = IT(JJ + 1) ** 2
```

Это необходимо, поскольку JJ в цикле DØ2J ограничивает поиск делителей среди найденных простых чисел.

Посмотрим на примере, как это происходит: вначале NCAR = 9; JJ = 2; I = 4;

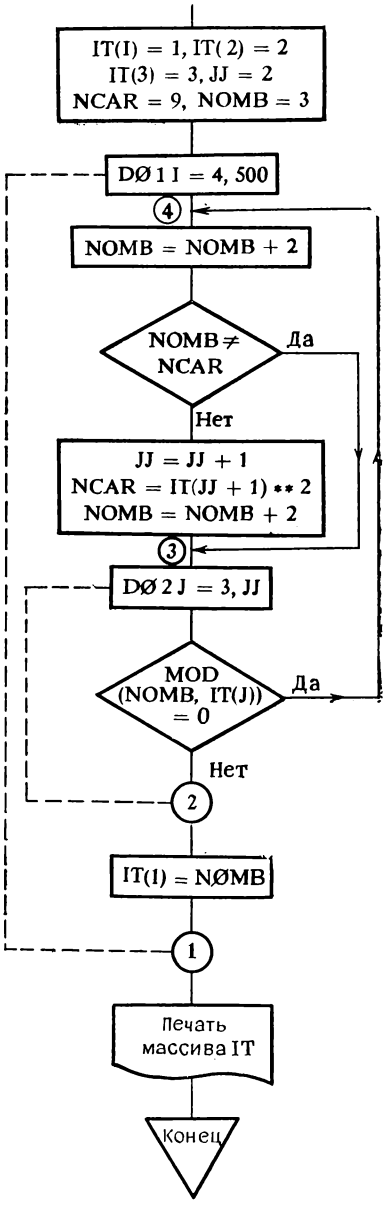


Рис. 53

```
C       ВЫЧИСЛЕНИЕ  500 ПЕРВЫХ  ПРОСТЫХ ЧИСЕЛ  .
        DIMENSION IT(500)
        IT(1)=1
        IT(2)=2
        IT(3)=3
        JJ=2
        NCAR=9
        NOMB=3
        DO 1 I=4,500
4       NOMB=NOMB+2
        IF(NOMB.NE.NCAR) GO TO 3
        JJ=JJ+1
        NCAR=IT(JJ+1)**2
        NOMB=NOMB+2
3       DO 2 J=3,JJ
        IF(MOD(NOMB,IT(J)).EQ.0) GO TO 4
2       CONTINUE
1       IT(I)=NOMB
        PRINT 100,IT
100     FORMAT(15I6)
      STOP 1
      END
```

```
   1    2    3    5    7   11   13   17   19   23   29   31   37   41   43
  47   53   59   61   67   71   73   79   83   89   97  101  103  107  109
 113  127  131  137  139  149  151  157  163  167  173  179  181  191  193
 197  199  211  223  227  229  233  239  241  251  257  263  269  271  277
 281  283  293  307  311  313  317  331  337  347  349  353  359  367  373
 379  383  389  397  401  409  419  421  431  433  439  443  449  457  461
 463  467  479  487  491  499  503  509  521  523  541  547  557  563  569
 571  577  587  593  599  601  607  613  617  619  631  641  643  647  653
 659  661  673  677  683  691  701  709  719  727  733  739  743  751  757
 761  769  773  787  797  809  811  821  823  827  829  839  853  857  859
 863  877  881  883  887  907  911  919  929  937  941  947  953  967  971
 977  983  991  997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063
1069 1087 1091 1093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181
1187 1193 1201 1213 1217 1223 1229 1231 1237 1249 1259 1277 1279 1283 1289
1291 1297 1301 1303 1307 1319 1321 1327 1361 1367 1373 1381 1399 1409 1423
1427 1429 1433 1439 1447 1451 1453 1459 1471 1481 1483 1487 1489 1493 1499
1511 1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 1609
1613 1619 1621 1627 1637 1657 1663 1667 1669 1693 1697 1699 1709 1721 1723
1733 1741 1747 1753 1759 1777 1783 1787 1789 1801 1811 1823 1831 1847 1861
1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 1949 1951 1973 1979
1987 1993 1997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081 2083
2087 2089 2099 2111 2113 2129 2131 2137 2141 2143 2153 2161 2179 2203 2207
2213 2221 2237 2239 2243 2251 2267 2269 2273 2281 2287 2293 2297 2309 2311
2333 2339 2341 2347 2351 2357 2371 2377 2381 2383 2389 2393 2399 2411 2417
2423 2437 2441 2447 2459 2467 2473 2477 2503 2521 2531 2539 2543 2549 2551
2557 2579 2591 2593 2609 2617 2621 2633 2647 2657 2659 2663 2671 2677 2683
2687 2689 2693 2699 2707 2711 2713 2719 2729 2731 2741 2749 2753 2767 2777
2789 2791 2797 2801 2803 2819 2833 2837 2843 2851 2857 2861 2879 2887 2897
2903 2909 2917 2927 2939 2953 2957 2963 2969 2971 2999 3001 3011 3019 3023
3037 3041 3049 3061 3067 3079 3083 3089 3109 3119 3121 3137 3163 3167 3169
3181 3187 3191 3203 3209 3217 3221 3229 3251 3253 3257 3259 3271 3299 3301
3307 3313 3319 3323 3329 3331 3343 3347 3359 3361 3371 3373 3389 3391 3407
3413 3433 3449 3457 3461 3463 3467 3469 3491 3499 3511 3517 3527 3529 3533
3539 3541 3547 3557 3559
```

Рис. 54

NØMB = 3. Переменная NØMB принимает значение 5, которое не равно NCAR. Следовательно, происходит переход к метке 3, выполняется DØ-цикл для J от 3 до 2 (т. е. всего один раз для 3). Число 3 не является делителем 5, и поэтому 5 — простое число.

Переменная I принимает значение 5; NØMB становится равной 7. Значение NØMB опять не равно NCAR. И снова мы выполняем цикл DØ2 для одного значения J = 3. Число 3 не является делителем 7, следовательно, 7 — простое число.

Теперь I принимает значение 6. Переменная NØMB принимает значение 9, которое в точности равно NCAR. Бессмысленно идти дальше, поскольку известно, что 9 — не простое число и нужно модифицировать переменную JJ (которая примет значение 3) и NCAR, которая станет равной $IT(3+1)^2$, т. е. 25 и т. д.

Кажущаяся сложность этого решения вполне окупается. Так, если на предыдущее решение требуется 2780 мсек, то в данном случае достаточно 200 мсек (не учитывается время, затрачиваемое на печать).

3.3. ПРОСТЫЕ ЧИСЛА: РЕШЕТО ЭРАТОСФЕНА

(*условие задачи на стр.* 34)

Мы приведем два решения; первое напрашивается само собой, во втором решении делается попытка уменьшить время вычисления.

Мы получим первое решение, буквально следуя условию задачи и не прибегая к каким-либо арифметическим ухищрениям. По определению нам известно, что простые числа — это те числа, которые делятся только на себя и на единицу.

Таким образом, первое, что нужно сделать, это заполнить массив T последовательностью, содержащей 3000 первых целых чисел. Соответствующие действия выполняются в цикле DØ1 (рис. 55 и 56).

Чтобы ограничить поиск, нам необходимо знать самый большой точный квадрат, меньший 3000. Для этого достаточно извлечь квадратный корень из этого числа и сохранить только целую часть полученного результата, т. е. выполнить инструкцию K = SQRT(3000.). Таким образом можно написать

```
      |  DØ 2 I=1, 100
      |  IF (I**2.GT.3000) GØTØ 3
      |  K=I**2
    2 |  CØNTINUE
    3 |  здесь K искомая величина
```

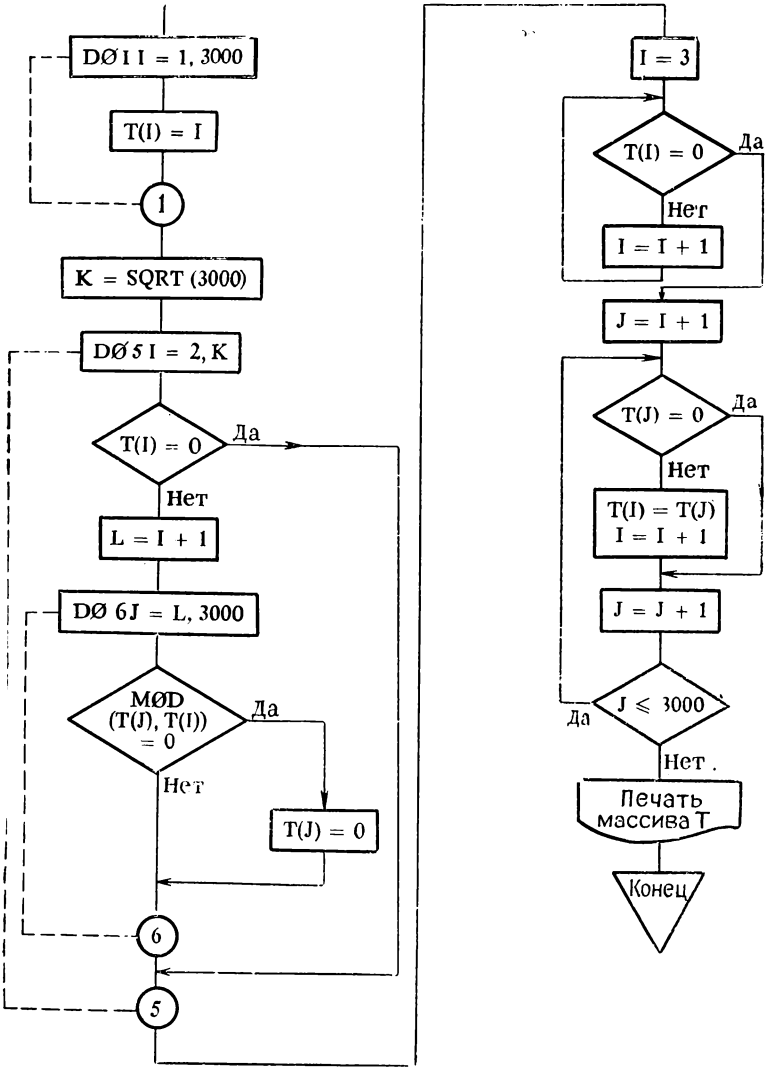

DØ 1 I = 1, 3000
T(I) = I
1
K = SQRT (3000)
DØ 5 I = 2, K
T(I) = 0
Да
Нет
L = I + 1
DØ 6 J = L, 3000
MØD (T(J), T(I)) = 0
Да
Нет
T(J) = 0
6
5
I = 3
T(I) = 0
Да
Нет
I = I + 1
J = I + 1
T(J) = 0
Да
Нет
T(I) = T(J)
I = I + 1
J = J + 1
J ⩽ 3000
Да
Нет
Печать массива T
Конец

Рис. 55

```
C       ПРОСТЫЕ ЧИСЛА МЕНЬШЕ 3000(РЕШЕТО ЭРАТОСФЕНА)
C
      INTEGER T(3000)
      DO 1 I=1,3000
    1 T(I)=I
      K=SQRT(3000.)
      DO 5 I=2,K
      IF(T(I).EQ.0) GO TO 5
      L=I+1
      DO 6 J=L,3000
      IF (MOD(T(J),T(I)).EQ.0) T(J)=0
    6 CONTINUE
    5 CONTINUE
      I=3
    7 IF(T(I).EQ.0) GO TO 8
      I=I+1
      GO TO 7
    8 J=I+1
   10 IF(T(J).EQ.0) GO TO 9
      T(I)=T(J)
      I=I+1
    9 J=J+1
      IF(J.LE.3000) GO TO 10
      N=I-1
      PRINT 11,(T(K),K=1,N)
   11 FORMAT(15I6)
      STOP
      END
```

1	2	3	5	7	11	13	17	19	23	29	31	37	41	43
47	53	59	61	67	71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167	173	179	181	191	193
197	199	211	223	227	229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347	349	353	359	367	373
379	383	389	397	401	409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523	541	547	557	563	569
571	577	587	593	599	601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727	733	739	743	751	757
761	769	773	787	797	809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937	941	947	953	967	971
977	983	991	997	1009	1013	1019	1021	1031	1033	1039	1049	1051	1061	1063
1069	1087	1091	1093	1097	1103	1109	1117	1123	1129	1151	1153	1163	1171	1181
1187	1193	1201	1213	1217	1223	1229	1231	1237	1249	1259	1277	1279	1283	1289
1291	1297	1301	1303	1307	1319	1321	1327	1361	1367	1373	1381	1399	1409	1423
1427	1429	1433	1439	1447	1451	1453	1459	1471	1481	1483	1487	1489	1493	1499
1511	1523	1531	1543	1549	1553	1559	1567	1571	1579	1583	1597	1601	1607	1609
1613	1619	1621	1627	1637	1657	1663	1667	1669	1693	1697	1699	1709	1721	1723
1733	1741	1747	1753	1759	1777	1783	1787	1789	1801	1811	1823	1831	1847	1861
1867	1871	1873	1877	1879	1889	1901	1907	1913	1931	1933	1949	1951	1973	1979
1987	1993	1997	1999	2003	2011	2017	2027	2029	2039	2053	2063	2069	2081	2083
2087	2089	2099	2111	2113	2129	2131	2137	2141	2143	2153	2161	2179	2203	2207
2213	2221	2237	2239	2243	2251	2267	2269	2273	2281	2287	2293	2297	2309	2311
2333	2339	2341	2347	2351	2357	2371	2377	2381	2383	2389	2393	2399	2411	2417
2423	2437	2441	2447	2459	2467	2473	2477	2503	2521	2531	2539	2543	2549	2551
2557	2579	2591	2593	2609	2617	2621	2633	2647	2657	2659	2663	2671	2677	2683
2687	2689	2693	2699	2707	2711	2713	2719	2729	2731	2741	2749	2753	2767	2777
2789	2791	2797	2801	2803	2819	2833	2837	2843	2851	2857	2861	2879	2887	2897
2903	2909	2917	2927	2939	2953	2957	2963	2969	2971	2999				

Рис. 56

И все-таки нелепо писать четыре инструкции там, где достаточно одной; более того, в этом цикле имеются два возведения в квадрат, которые компилятор обычно переводит одним простым умножением. Можно было бы записать эту группу инструкций несколько иначе, оставив лишь одно возведение в квадрат. Но в любом случае, по-видимому, бесполезно задерживаться на том, что было бы возможно, после того, как выбор и вполне оправданный был сделан.

Теперь мы исключим все числа, кратные 2, которое заведомо является первым простым числом (оно делится только на себя и на 1), и будем так продолжать вплоть до K. Эти действия выполняются в цикле DØ 5 I, т. е. рассматриваются T(J) при всех значениях J между I + 1 и 3000, и всякий раз, когда находится число, кратное T(I), оно заменяется нулем. Когда I увеличивается на 1, может случиться, что попадется уже исключенное число, что приведет к делению на 0 и сразу же к немедленному прекращению вычислений. Следовательно, прежде, чем начать цикл DØ6J, необходимо убедиться, что T(I) отлично от нуля. Если оно равно 0, то переходим к следующему T(I).

Пример:

```
Исходный массив : 1 2 3 4 5 6 7 8 9  10 11 12 ...
После исключения
чисел, кратных 2 : 1 2 3 0 5 0 7 0 9  0  11 0  ... I=2
После исключения
чисел, кратных 3 : 1 2 3 0 5 0 7 0 0  0  11 0  ... I=3
```

Нетрудно видеть, что произошло бы, если бы при I = 4 не было проверки T(I) на равенство 0!

Цикл DØ6 необходимо начинать с I + 1, а не с I, иначе всякий раз будет исключаться T(I), что приведет к ошибке.

Чтобы узнать, делится ли T(J) на T(I), достаточно сравнить с нулем остаток от деления T(J) на T(I); функция MØD, которая вычисляет остаток, именно для этого и предназначена.

Когда J пробежит от L до 3000, переменная I в цикле DØ5 примет следующее значение и цикл по I повторится, и так до тех пор, пока I не превысит K, после чего произойдет выход из цикла 5.

Чтобы напечатать таблицу простых чисел в удобном виде, полезно сгруппировать все числа, не равные 0 в левую часть массива. Перед этой операцией массив T(I) выглядит следующим образом:

```
1 2 3 0 5 0 7 0 0 0 11 0 13 0 0 0 17 0 19 ...
```

и нужно получить

1 2 3 5 7 11 13 17 19 ...

Прежде всего необходимо найти первое T(I), равное нулю, начиная поиск с T(3), поскольку достоверно известно, что T(2) не было исключено. Здесь не используется DØ-цикл, так как бесполезно проверять, что индекс не превосходит заранее фиксированного конечного значения, известно только, что между 2 и 3000 имеются простые числа. Четыре инструкции, начинающиеся с I = 3 и предшествующие метке 8, позволяют вычислить индекс I первого нулевого элемента. Далее происходит поиск первого ненулевого элемента T(J), и он помещается на место T(I). Заметим, что бессмысленно «готовить» место, где было простое число, полагая T(J) = 0, потому что все числа, которые еще предстоит переместить, расположены справа и со временем они сотрут прежние значения. Достаточно того, что часть массива T, лежащая правее наибольшего из найденных простых чисел, не печатается.

Всякий раз, когда T(J) заносится в T(I), необходимо увеличить J *и* I. В противном случае нужно увеличить только J. Значение J изменяется быстрее, чем I, поэтому, когда оно превысит 3000, необходимо остановиться. В результате первые I — 1 элементов в массиве T будут ненулевыми, поскольку I указывает на первый нулевой элемент. Для печати используется неявный цикл.

ВОПРОС:

Почему в касающихся DØ-циклах используются две различные метки 5 и 6?

ОТВЕТ:

Потому что во внешнем цикле есть инструкция GØTØ, которая предписывает пропустить все инструкции цикла, включая внутренний цикл. Некоторые компиляторы (например, компилятор для машин IBM) не обнаруживают эту ситуацию и без какого-либо диагностического сообщения порождают программу, которая зацикливается.

Другое решение

Немного подумав, можно уменьшить время работы программы в отношении 1 к 4 и найти все простые числа, меньшие 5997 при том же размере массива T(3000).

Примем как известное, что четные числа, за исключением 2, не простые, поэтому не будем их заносить в массив T, который теперь будет содержать только (кроме 2) нечетные числа (цикл DØ1 на рис. 57 и 58).

Первым как кратное M исключается число M^2 (смотри учебник по арифметике). Затем, очевидно, исключаются все последующие числа с шагом M. Но в нашем массиве нет чет-

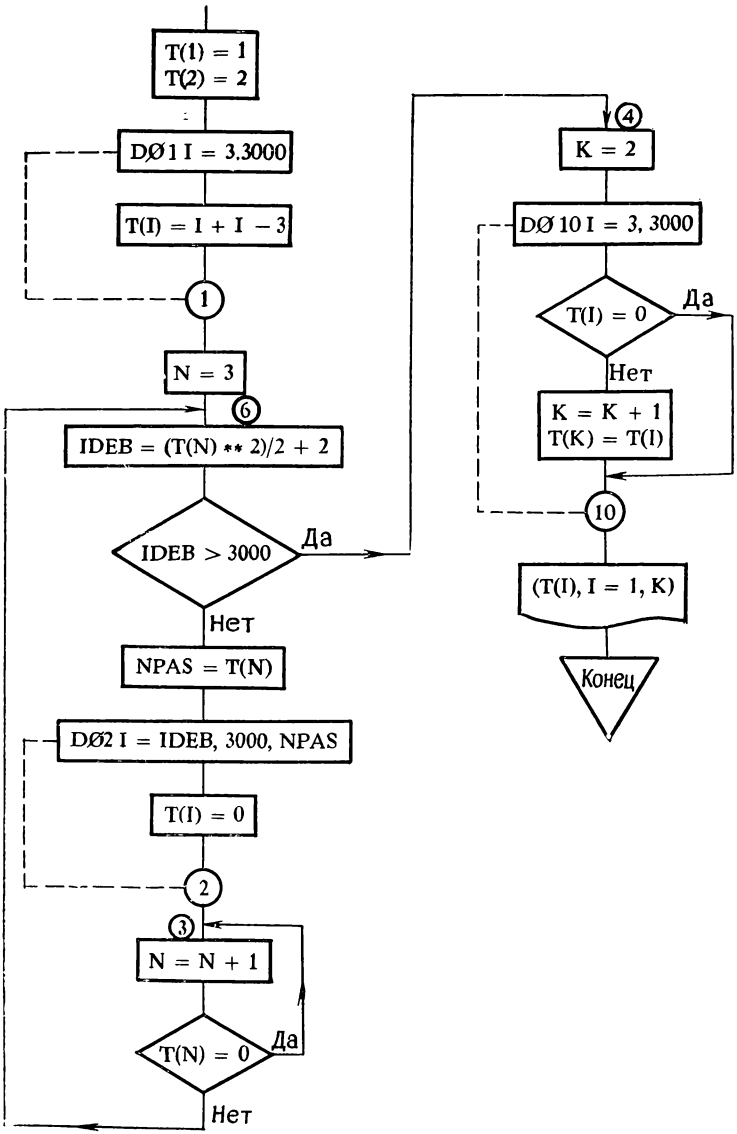

T(1) = 1
T(2) = 2
DØ1 I = 3,3000
T(I) = I + I − 3
1
N = 3
6
IDEB = (T(N) ** 2)/2 + 2
IDEB > 3000
Да
Нет
NPAS = T(N)
DØ2 I = IDEB, 3000, NPAS
T(I) = 0
2
3
N = N + 1
T(N) = 0
Да
Нет
4
K = 2
DØ 10 I = 3, 3000
T(I) = 0
Да
Нет
K = K + 1
T(K) = T(I)
10
(T(I), I = 1, K)
Конец

Рис. 57

ных чисел, поэтому придется вычислять адрес элемента в массиве T, где находится интересующее нас значение.

Следовательно, значением IDEB будет $M^2/2$ (массив T содержит только одно значение из двух) плюс 2, поскольку уже

```
C     ПРОСТЫЕ ЧИСЛА (РЕШЕТО ЭРАТОСФЕНА)
C
            INTEGER T(3000)
            T(1)=1
            T(2)=2
            DO 1 I=3,3000
1           T(I)=I+I-3
            N=3
6           IDEB=(T(N)*T(N))/2+2
            IF (IDEB.GT.3000) GO TO 4
            NPAS=T(N)
            DO 2 I=IDEB,3000,NPAS
2           T(I)=0
3           N=N+1
            IF(T(N).EQ.0) GO TO 3
            GO TO 6
4           K=2
            DO 10 I=3,3000
            IF(T(I).EQ.0) GO TO 10
            K=K+1
            T(K)=T(I)
10          CONTINUE
            PRINT 100,K,(T(I),I=1,K)
100         FORMAT(I100//,(15I6))
            STOP 3
            END
```

Рис. 58

известно два числа. Возьмем, например, число M = 5, которое находится в T(4). Значение N = 4, а значение IDEB = 14.

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
M	1	2	3	5	7	~~9~~	11	13	~~15~~	17	19	~~21~~	23	25 ↑	~~27~~	29

Само собой разумеется, когда значение IDEB превысит 3000, все оставшиеся в массиве числа будут простыми. Пока этого не произошло, рассматривается массив T, начиная с с IDEB до 3000, с шагом, равным T(N) (как и в предыдущем решении), и всем элементам массива, адреса которых определяются таким образом, присваиваются нулевые значения. После выхода из этого цикла делается переход к следующему, еще не исключенному числу в массиве T (то есть к простому числу) и затем исключаются все кратные ему числа. Значение N увеличивается на 1, и, если T(N) отлично от 0, вычисления повторяются.

Уплотнение массива и затем его печать производятся в сущности так же, как и в предыдущем решении, но используется более простая и более компактная запись.

ВОПРОСЫ:

1. В двух рассмотренных решениях используются различные инструкции FØRMAT, в соответствии с которыми печатаются найденные простые числа. Почему пишется «I100», хотя доподлинно известно, что целые числа в машине никогда не содержат более десяти цифр?

2. Почему в той же инструкции FØRMAT 15I6 заключено в скобки?

ОТВЕТЫ:

1. На размер поля не налагается никаких других ограничений, кроме числа позиций в одной строке. Слева неиспользуемые позиции заполняются пробелами, поэтому можно было написать 95X, I5.

2. Если скобки опустить, то формат повторялся бы с самого начала, т. е. с I100, что привело бы к нарушению расположений чисел на бумаге. Таким образом, скобки предписывают повторение формата на уровне (15I6).

3.4. СОВЕРШЕННЫЕ ЧИСЛА

(*условие задачи на стр.* 35)

Вычисление организовано следующим образом:

- Переменная N принимает значения от 4 до 10 000, и среди них ведется поиск совершенных чисел.
- Переменная I принимает значения от 1 до N/2, которые рассматриваются как последовательность возможных делителей N. Ограничение N/2 очевидно, поскольку, если имеется какой-либо другой делитель, кроме 1, он не меньше 4 и не больше N/2.
- Всякий раз, когда обнаруживается, что I делит N, значение I записывается в элемент массива L с индексом IL (значение IL первоначально равно 0 и увеличивается на 1 перед каждой записью в массив). В этот же момент изменяется значение переменной K, которая соответствует сумме делителей.
- Как только значение I превысило N/2, производится сравнение N и K. Если их значения равны, то N — совершенное число и оно печатается одновременно со всеми IL делителями. Именно этот алгоритм и показан на блок-схеме (рис. 61), которой соответствует программа, представленная на рис. 59.

Для получения столь скудных результатов (рис. 60) затрачивается весьма внушительное время [1]). Возникает вопрос, как же улучшить этот метод.

Решение, которое мы только что рассмотрели, представляет собой типичный пример объединения тупости (в прин-

```
C          ВЫЧИСЛЕНИЕ СОВЕРШЕННЫХ ЧИСЕЛ
C
           DIMENSION L(100 )
           DO 1 N=4,10000
           IL=0
4          FORMAT(/,I5,3OI4)
           M=N/2
           K=0
           DO 2 I=1,M
           IF (MOD(N,I).NE.0) GO TO 2
           K=K+I
           IL=IL+1
           L(IL)=I
2          CONTINUE
           IF (K.NE.N) GO TO 1
3          PRINT 4, N, (L(J),J=1,IL)
1          CONTINUE
           STOP
       END
```

Рис. 59

ципе присущей вычислительной машине) с недостаточной сообразительностью (если не сказать леностью) некоторых программистов. Обобщение такого подхода к решению задач может объяснить столь безосновательную перегруженность некоторых вычислительных центров.

```
   6   1   2   3
  28   1   2   4   7  14
 496   1   2   4   8  16  31  62 124 248
8128   1   2   4   8  16  32  64 127 254 508101620324064
```

Рис. 60

Чтобы получить эффективное решение, необходимо целенаправленно ограничить число проверок, основываясь на простых арифметических соображениях. Следующие два момента вполне заслуживают нашего внимания.

1. Очевидно, что, как только значение K (сумма делителей числа N) превысит значение N ($I < N/2$), нет больше смысла продолжать проверку, поскольку K уже никогда не будет

[1]) 476 с на машине IBM 360/65.

равно N, т. е. можно перейти к проверке следующего значения N. Могут возразить, отметив, что дополнительная проверка, которая повторяется для каждого вновь найденного делителя, увеличит время вычислений. Арифметические соображения, которые вряд ли стоит здесь приводить, или проще эксперименты на некоторых числовых значениях могут свидетельствовать об обоснованности нашего предложения (полученное вычислительное время послужит тому убедительным подтверждением).

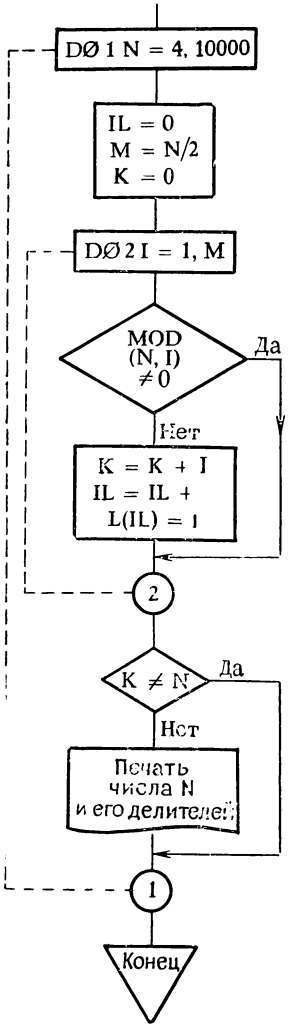


Рис. 61

2. Если при проверке делимости N на J остаток оказался нулевым, то одновременно получено два делителя числа N (частное и делитель), которые сразу же можно занести в массив L. Кроме того, из этого факта вытекает, что, когда частное становится меньше делителя, все возможные делители уже найдены и занесены в массив, и если их сумма не равна N, то нужно перейти к проверке другого значения N.

Итак, перепишем программу заново с учетом высказанных замечаний (рис. 62 и 63). Поскольку необходимо сохранить остаток и частное, нельзя использовать функцию MØD. Хотя внутренние схемы вычислительной машины позволяют одновременно получить остаток и частное от деления, в языке Фортран не существует функции, с помощью которой можно иметь их сразу же, и поэтому придется написать две инструкции с одинаковыми операндами.

Цикл DØ2I более не ограничивается значением N/2. Он просто исчезает и заменяется явной неконтролируемой прогрессией I, поскольку проверки, организованные так, как говорилось выше, приведут к концу поиска делителей раньше, чем I достигнет значения N/2. Следовательно, бесполезно производить сравнение при каждом повторении, как это делается в DØ-цикле.

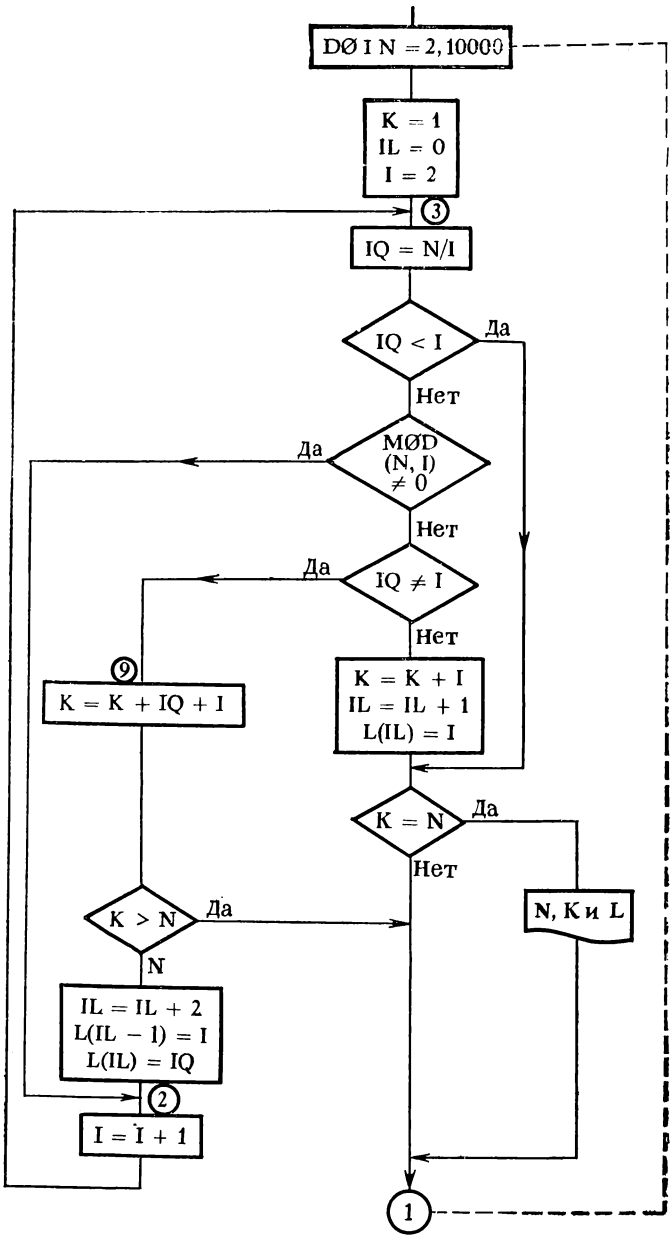

DØ I N = 2,10000
K = 1
IL = 0
I = 2
3
IQ = N/I
IQ < I
Да
Нет
MØD (N, I) ≠ 0
Да
Нет
IQ ≠ I
Да
Нет
9
K = K + IQ + I
K = K + I
IL = IL + 1
L(IL) = I
K = N
Да
Нет
K > N
Да
N
N, K и L
IL = IL + 2
L(IL − 1) = I
L(IL) = IQ
2
I = I + 1
1

Рис. 62

IQ — частное от деления N на I. Остаток вычисляется (с помощью функции MØD) только тогда, когда IQ > I. В противном случае изменяется значение N. Когда получается ненулевой остаток от деления на I, I не является делителем N и, следовательно, необходимо перейти к другому значению I, которое получается в инструкции I = I + 1 (с меткой 2). Если

```
      DIMENSION L(50)
      L(1)=1
      DO 1 N=2,10000
      K=1
      IL=1
      I=2
3     IQ=N/I
      IF(IQ.LT.I) GO TO 10
      IF(MOD(N,I).NE.0) GO TO 2
      IF(IQ.NE.I) GO TO 9
      K=K+I
      IL=IL+1
      L(IL)=I
      GO TO 10
9     K=K+IQ+I
      IF(K.GT.N) GO TO 1
      IL=IL+2
      L(IL-1)=I
      L(IL)=IQ
2     I=I+1
      GO TO 3
10    IF(K.NE.N) GO TO 1
      PRINT 100,N,(L(I),I=1,IL)
100   FORMAT(/(1516))
1     CONTINUE
      STOP
      END
```

```
   6    1    2    3
  28    1    2   14    4    7
 496    1    2  248    4  124    8   62   16   31
8128    1    2 4064    4 2032    8 1016   16  508   32  254   64  127
```

Рис. 63

I делит N, то в массив делителей L записывается I и IQ. Значение IL (индекс элемента массива L) изменяется таким образом, чтобы оно, как и ранее, указывало число известных элементов массива L в любой момент времени.

Нетрудно заметить, что наибольшее из значений I будет близким к $\sqrt{N}$, а $\sqrt{N}$ всегда меньше или равен N/2.

Печать результатов производится обычным образом, но мы должны отметить, что делители печатаются не в порядке возрастания величин, а в том порядке, в каком они расположились в массиве L. Впрочем, это не противоречит условию задачи. Стремясь к совершенству, можно было бы перед печатью обратиться к подпрограмме сортировки.

Теперь решение получается всего лишь за 19 с на той же машине, на которой ранее требовалось 476 с.

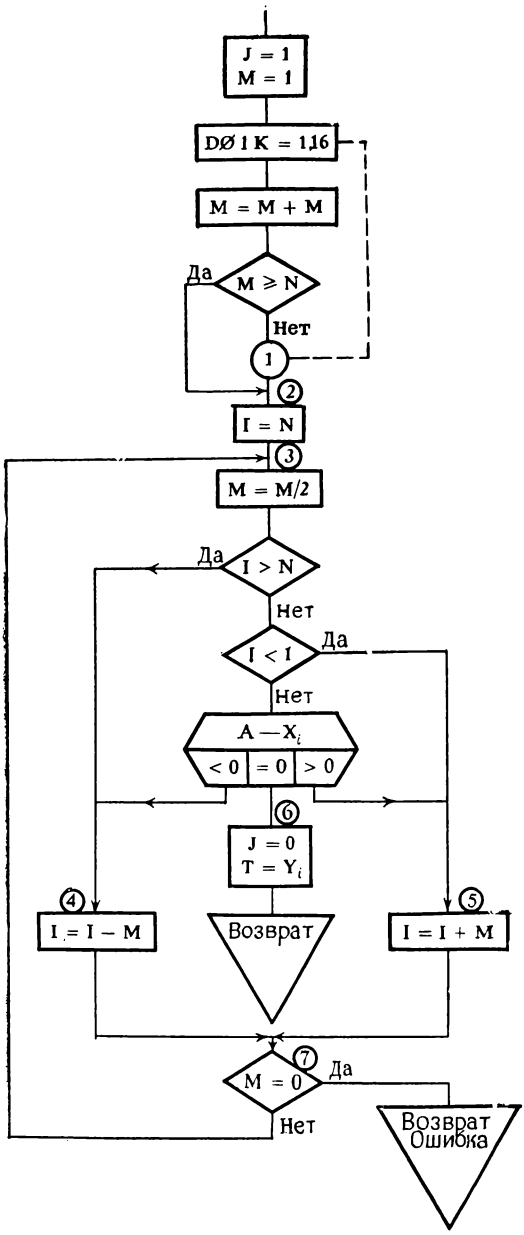

J = 1
M = 1
DØ 1 K = 1,16
M = M + M
Да
M ≥ N
Нет
1
2
I = N
3
M = M/2
Да
I > N
Нет
I < 1
Да
Нет
A — Xi
< 0
= 0
> 0
6
J = 0
T = Yi
4
I = I — M
Возврат
5
I = I + M
7
M = 0
Да
Нет
Возврат
Ошибка

Рис. 64

3.5. ДИХОТОМИЧЕСКИЙ ПОИСК В ТАБЛИЦЕ

(*условие задачи на стр.* 35)

Начнем с вычисления K. Полагая вначале $M = 2$, будем удваивать значение M при каждом повторении цикла DØ1. Когда значение M станет больше N, произойдет выход из цикла. В этот момент $M = 2^{K+1}$. Таким образом, потребуется $K + 1$ повторений цикла, и, следовательно, нужно разделить M на 2 (см. блок-схему на рис. 64 и программу на рис. 65).

Предполагается, что в таблице не более 65 536 элементов (2^{16}) и не менее 2 элементов.

```
C       ПОИСК В МАССИВЕ
C
        FUNCTION TABLE (A,X,Y,N,J)
        DIMENSION X(1), Y(1)
        J=1
        M=1
        DO 1 K=1,16
        M=M+M
        IF (M.GE.N) GO TO 2
1       CONTINUE
2       I=N
3       M=M/2
        IF (I.GT.N) GO TO 4
        IF (I.LT.1) GO TO 5
        IF (A-X(I))4, 6, 5
4       I=I-M
        GO TO 7
5       I=I+M
7       IF (M.EQ.0) RETURN
        GO TO 3
6       TABLE=Y(I)
        J=0
        RETURN
        END
```

Рис. 65

Начальное значение I равно N. Заметим, что на самом деле можно было бы выбрать любое начальное значение, лежащее между 1 и N. Вместо DØ-цикла по значению K проще проверить M на нуль и таким образом определить момент, когда исчерпаны все возможности поиска в таблице.

Затем необходимо убедиться, что I попадает в заданные пределы. В первый раз эта проверка, очевидно, бесполезна, но она совершенно необходима при всех других повторениях. Если I лежит в заданных пределах, то рассматривается соотношение между X и A. Значение I корректируется требуемым образом, изменяется M, и проверки повторяются при новом значении I. Обычный выход из цикла определенно свидетельствует о том, что решения нет, и происходит возврат со зна-

чением J, равным 1 (это значение J принимает в самом начале программы). В противном случае J присваивается значение 0, а TABLE — значение Y(I).

Можно было бы использовать цикл DØ по J, изменяющемуся от 1 до K + 1, и при каждом повторении вычислять 2^{K+1-J}, но в этом случае программа работала бы значительно дольше. Так как имеется переменная $M = 2^{k+1}$, простым делением на 2 можно получить ближайшую меньшую степень 2, вплоть до нулевой (результат от деления 1/2).

ВОПРОСЫ:

1. Почему в цикле DØ1 для вычисления M используется сложение (M + M), а не умножение (2 * M)?
2. Почему в качестве размера массивов X и Y указана 1, а не N?

ОТВЕТЫ:

1. Потому что сложение выполняется быстрее умножения.
2. Для одномерных массивов это неважно, но в некоторых компиляторах излишне... простых отдается предпочтение этой форме описания массива.

3.6. ПОИСК НАИБОЛЬШЕГО ЭЛЕМЕНТА В ТРЕХМЕРНОМ МАССИВЕ

(*условие задачи на стр.* 37)

```
C       ПОИСК НАИБОЛЬШЕГО ЧИСЛА В ТРЕХМЕРНОМ МАССИВЕ
C
        DIMENSION T(3,5,7)
C
C       ПОДГОТОВКА МАССИВА ДЛЯ ПРОВЕРКИ
        DO 100 I=1,3
        DO 100 J=1,5
        DO 100 K=1,7
100     T(I,J,K)=2*I+5*J-K
        PRINT 200,T
200     FORMAT (15F6.0)
C
        GR=T(1,1,1)
        DO 1 I=1,3
        DO 1 J=1,5
        DO 1 K=1,7
1       GR=AMAX1(GR,T(I,J,K))
        PRINT 2, GR
2       FORMAT (//E15.7)
        STOP
        END
```

Рис. 66

Предыдущее решение (рис. 66) имеет то достоинство, что оно самоочевидное и наглядное, но программа получилась довольно медленной, поскольку на вычисление адреса при

трех индексах требуется время. Чтобы улучшить это решение[1]), нужно использовать инструкцию EQUIVALENCE, позволяющую уменьшить число индексов массива T до одного (программа на рис. 67).

```
C       ПОИСК НАИБОЛЬШЕГО ЧИСЛА В ТРЕХМЕРНОМ МАССИВЕ
C             --ВТОРОЕ РЕШЕНИЕ(ОПТИМИЗИРОВАННОЕ)--
C
        DIMENSION T(3,5,7), R(105)
        EQUIVALENCE (T(1,1,1),R(1))
C
C       ПОДГОТОВКА МАССИВА ДЛЯ ПРОВЕРКИ
        DO 100 I=1,3
        DO 100 J=1,5
        DO 100 K=1,7
100     T(I,J,K)=2*I+5*J-K
        PRINT 200,T
200     FORMAT (15F6.0)
C
        GR=R(1)
        DO 1 I=2,105
1       GR=AMAX1(R(I),GR)
        PRINT 2, GR
2       FORMAT (//E15.7)
        STOP
        END
```

Рис. 67

Заметим, что в обоих случаях используется функция AMAX1, которая позволяет избежать инструкции IF вида

$$\text{IF (R (I).GT.GR) GR} = \text{R (I)}$$

Однако на уровне команд, генерируемых компилятором, разницы почти нет.

3.7. ПОИСК ДВУХ ОДИНАКОВЫХ ЧИСЕЛ В ДВУМЕРНОМ МАССИВЕ

(*условие задачи на стр.* 37)

И опять-таки простое решение (рис. 68) неудовлетворительно, поскольку программа излишне медленна. Не спасает даже некоторое улучшение, полученное за счет разделения двух условий и *предварительной* проверки двух чисел, одно из которых присваивается простой переменной B, чтобы избежать многократного вычисления индексов A(I, J) (в случае компиляторов, в которых не предусмотрена оптимизация).

[1]) Количественно первое и второе решения на машине БЭСМ-6 характеризуются так: БЭСМ-6/Д (1,9 и 1,8 с), БЭСМ-6/М (0,24 и 0,2 с). Массив был задан иначе, а именно T (7, 32, 50). — *Прим. ред.*

Если в первой инструкции IF обнаруживается два одинаковых числа, нужно убедиться, что их индексы не равны, дабы не сравнивать число само с собой.

```
      DIMENSION A(30,7)
C
C     ПОДГОТОВКА МАССИВА
      DO 10 I=1,30
      DO 10 J=1,7
10    A(I,J)=100*I+J
      A(23,4)=A(11,6)
C
C     СОБСТВЕННО РЕШЕНИЕ
      DO 2 I=1,30
      DO 2 J=1,7
      B=A(I,J)
      DO 2 K=1,30
      DO 2 L=1,7
      IF (A(K,L).NE.B) GO TO 2
      IF (K.EQ.I.AND.L.EQ.J) GO TO 2
      GO TO 1
2     CONTINUE
1     PRINT 3, I, J, K, L, B
3     FORMAT (4I3,E15.7)
      STOP
      END
```

Рис. 68

```
      DIMENSION A(30,7),C(210)
C
C     ПОДГОТОВКА МАССИВА
      DO 10 I=1,30
      DO 10 J=1,7
10    A(I,J)=100*I+J
      A(23,4)=A(11,6)
C
      EQUIVALENCE(A(1,1),C(1))
      DO 1 I=1,209
      K=I+1
      D=C(I)
      DO 1 J=K,210
      IF(D.EQ.C(J)) GO TO 2
1     CONTINUE
2     I1=MOD(I,30)
      I2=MOD(J,30)
      J1=I/30+1
      J2=J/30+1
      PRINT 3,I1,J1,I2,J2,D
3     FORMAT(4I3,E15.7)
      STOP
      END
```

Рис. 69

В условии задачи сказано, что в массиве имеются только два одинаковых числа и поэтому никогда не бывает нормального выхода из DØ-цикла. Следовательно, при печати индексы I, J, K и L определены.

Другое решение

При оптимизации этого решения можно подумать об использовании эквивалентности с одномерным массивом C, имеющим $30 \times 7 = 210$ элементов. Применение инструкции EQUIVALENCE возможно, но в условии задачи требуется указать индексы элементов, а их придется перевычислять. Это не представит особого труда, поскольку известно, что массивы располагаются по столбцам: индексы (I1, J1) и (I2, J2) соотносятся с массивом A. Используется тот факт, что инструкция EQUIVALENCE (рис. 69) устанавливает следующее соответствие между массивами A и C:

$$\begin{array}{cccccccccc} A_{1,1} & A_{1,2} & A_{1,3} & \cdots & A_{1,7} & A_{2,1} & A_{2,2} & \cdots & A_{30,7} \\ C_1 & C_{31} & C_{61} & & C_{181} & C_2 & C_{32} & & C_{210} \end{array}$$

3.8. СУММИРОВАНИЕ ЭЛЕМЕНТОВ МАССИВА, СУММА ИНДЕКСОВ КОТОРЫХ РАВНА КОНСТАНТЕ

(*условие задачи на стр.* 37)

Поскольку формат описывает карту, мы можем воспользоваться любой из спецификаций: 10F8.0 или 10E8.0 (предпо-

```
C       СУММИРОВАНИЕ ЧЛЕНОВ С СУММОЙ ИНДЕКСОВ,РАВНОЙ КОНСТАНТЕ
C
        DIMENSION X(10,30)
        S=0
        READ 1, X
1       FORMAT (10F8.0)
        PRINT 8, ((X(I,J),J=1,30),I=1,10)
8       FORMAT(/15F6.0/15F6.0)
        READ 2, K
2       FORMAT (I2)
        IF (K.GT.1.AND.K.LT.41) GO TO 3
        PRINT 4, K
4       FORMAT (22H НЕВЕРНОЕ ЗНАЧЕНИЕ  K:, I3)
        STOP
C
3       DO 5 I=1,10
        J=K-I
        IF (J.LT.1.OR.J.GT.30) GO TO 5
        S=S+X(I,J)
5       CONTINUE
        PRINT 6, S
6       FORMAT (7H-SOMME=, E15.6)
        STOP
        END
```

Рис. 70

лагая, конечно, что десятичная точка перфорируется на карте). Тогда, не долго думая, можно написать программу, листинг которой представлен на рис. 70. В ней последовательно

просматриваются все возможные значения I и каждый раз вычисляется единственно возможное значение J. Если J оказывается вне допустимых пределов, делается переход к следующему значению I. В общей сложности требуется 10 повторений цикла. После того как мы убедимся, что программа функционирует нормально, следует подумать о том, как можно улучшить решение, и обратить внимание на пределы, в которых изменяется I. Наименьшее значение, очевидно, равно 1 (кроме случая, когда $K > 31$ [число столбцов в массиве

```
C       СУММИРОВАНИЕ ЧЛЕНОВ С СУММОЙ ИНДЕКСОВ, РАВНОЙ КОНСТАНТЕ
C            -- ОПТИМИЗИРОВАННОЕ РЕШЕНИЕ --
C
        DIMENSION X(10,30)
        S=0
        READ 1, X
1       FORMAT (10F8.0)
        PRINT 8, ((X(I,J),J=1,30),I=1,10)
8       FORMAT(/15F6.0/15F6.0)
        READ 2, K
2       FORMAT (I2)
        IF (K.GT.1.AND.K.LT.41) GO TO 3
        PRINT 4, K
4       FORMAT (22H НЕВЕРНОЕ ЗНАЧЕНИЕ  K:, I3)
        STOP
C
3       IMIN=MAX0(1,K-30)
        IMAX=MIN0(10,K-1)
        DO 5 I=IMIN, IMAX
5       S=S+X(I,K-I)
        PRINT 6, S
6       FORMAT (7H-SOMME=, E15.6)
        STOP
        END
```

Рис. 71

$+ 1$], и тогда оно равно $K - 30$). Это последнее выражение не следует вычислять, если $K < 31$, так как I получилось бы отрицательным или нулевым (при $K = 31$, $K - 30$ равно 1). Поэтому наименьшее значение I вычисляется следующим образом:

$$\text{MAX0}(1,\ K - 30)$$

Те же рассуждения показывают, что наибольшее значение K равно 10 (число строк в массиве), но если $K \leqslant 10$, то оно равно $K - 1$ и может быть получено с помощью выражения

$$\text{MIN0}(10,\ K - 1)$$

поскольку при значениях $K > 11$, $K - 1$ больше числа строк в массиве.

Теперь можно написать программу, листинг которой воспроизведен на рис. 71.

Можно показать, что программа правильна, выписав пределы изменения I и J:

$$1 \leqslant I \leqslant 10,$$

$$1 \leqslant J \leqslant 30.$$

Поскольку $I = K - J$, наименьшее значение I получится при наибольшем значении J, равном 30, а наибольшее значение I — при наименьшем значении J, равном 1. Отсюда

$$K - 30 \leqslant I \leqslant K - 1$$

и поскольку наименьшее из допустимых значений J равно 1, а наибольшее — 10, мы приходим к тому же результату.

3.9. КВАДРАТНЫЙ КОРЕНЬ

(*условие задачи на стр.* 37)

Здесь не используется массив для хранения Y_I и Y_{I+1}, и в этом нет ошибки, поскольку речь идет о вычислении с по-

```
C       КВАДРАТНЫЙ КОРЕНЬ ИЗ ВЕЩЕСТВЕННОГО ЧИСЛА
C
10      READ 1, A
1       FORMAT (E10.0)
        PRINT 20, A
20      FORMAT (1X, 3HA =, E15.7)
        IF (A.GE.0.) GO TO 3
        PRINT 2
2       FORMAT (1X, 15ОТРИЦ. ЗНАЧЕНИЕ /)
        GO TO 10
3       S=A
        IF (A.EQ.0) GO TO 4
5       T=(S+A/S)*.5
        IF (ABS((T-S)/T).LE.1.E-6) GO TO 4
        S=T
        GO TO 5
4       PRINT 6, S
6       FORMAT (9H КОРЕНЬ =, E15.7//)
        GO TO 10
        END
```

Рис. 72

мощью итераций, а не о вычислении ряда. Мы умышленно назвали переменные S и T, чтобы подчеркнуть этот факт (см. программу на рис. 72).

Разумеется, нельзя писать

$$1/2 * (S + A/S)$$

так как неминуемо получится нуль, поскольку 1 и 2 — целые числа, и частное от их деления равно нулю.

3.10. ВЫЧИСЛЕНИЕ e

(*условие задачи на стр.* 38)

Наиболее очевидное решение показано на рис. 73, а результаты, полученные при выполнении этой программы, на рис. 74.

```
      DOUBLE PRECISION E,F
      E=1
      DO 1 I=1,20
      F=1
      DO 2 J=1,I
2     F=F*J
      E=E+1/F
1     PRINT 3, I, E
      STOP
3     FORMAT (I3, D25.15)
      END
```

Рис. 73

```
 1   .2000000000000000D+01
 2   .2500000000000000D+01
 3   .2666666666666663D+01
 4   .2708333333333280+01
 5   .2716666666666580+01
 6   .2718055555555470+01
 7   .2718253968253880+01
 8   .2718278769841240+01
 9   .2718281525573120+01
10   .2718281801146310+01
11   .2718281826198460+01
12   .2718281828286110+01
13   .2718281828446730+01
14   .2718281828458180+01
15   .2718281828458940+01
16   .2718281828459010+01
17   .2718281828459010+01
18   .2718281828459010+01
19   .2718281828459010+01
20   .2718281828459010+01
```

Рис. 74

```
      DOUBLE PRECISION E, F
      E=1
      F=1
      DO 1 I=1,20
      F=F*I
      E=E+1/F
1     PRINT 3, I, E
      STOP
3     FORMAT (I3, D25.15)
      END
```

Рис. 75

Заметим, что цикл DØ2 можно исключить, поскольку в нем для каждого значения I повторяются все умножения. Лучше,

конечно, сохранять произведение, полученное при предыдущем значении I. Тогда можно обойтись только одним умножением. Программа становится такой, как на рис. 75.

Но можно сделать еще лучше и вообще исключить умножение, поскольку:

$$\frac{1}{n!} = \frac{1}{(n-1)!} \times \frac{1}{n} = \frac{1/(n-1)!}{n}.$$

Будем вычислять 1/n! вместо n!, что и реализовано в программе на рис. 76.

```
      DOUBLE PRECISION E, F
      E=1
      F=1
      DO 1 I=1,20
      F=F/I
      E=E+F
1     PRINT 3, I, E
      STOP
3     FORMAT (I3, D25.15)
      END
```

Рис. 76

3.11. ПОСЛЕДОВАТЕЛЬНОСТИ БЕЗ ПОВТОРЕНИЙ

(*условие задачи на стр.* 38)

В качестве первой цифры будем последовательно брать 0, 1, 2, ..., 9. Затем среди целых чисел от 0 до 9 возьмем

```
C     СОВЕРШЕННО РАЗЛИЧНЫЕ ЧИСЛА -
C
      DIMENSION N(5040)
      M=0
      DO 1 I=1,10
      DO 1 J=1,10
      IF (J.EQ.I) GO TO 1
      DO 1 K=1,10
      IF (K.EQ.J.OR.K.EQ.I) GO TO 1
      DO 1 L=1,10
      IF (L.EQ.K.OR.L.EQ.J.OR.L.EQ.I) GO TO 1
      M=M+1
      N(M)=1000*(I-1)+100*(J-1)+10*(K-1)+L-1
1     CONTINUE
      PRINT 2, N
2     FORMAT (20I6)
      STOP
      END
```

Рис. 77

первое попавшееся, которое не равно предыдущему. При подборе третьей цифры будем действовать точно так же, проверяя, что она отличается от *двух предыдущих*. Те же рассуж-

дения годятся и при выборе четвертой цифры. Таким образом, используя четыре DØ-цикла, можно получить приемлемое решение (рис. 77).

ВОПРОСЫ:

1. Почему I, J, K и L изменяются от 1 до 10, а не от 0 до 9? Казалось бы, что тем самым только усложняется вычисление N(M).

2. Можно ли увеличивать M с помощью DØ-цикла?

3. Можно ли выражение, значение которого присваивается N(M), записать проще?

ОТВЕТЫ:

1. Потому что существуют компиляторы, при разработке которых почтительно относились к стандартам, не позволяющим начинать DØ-цикл с 0.

2. Нет, поскольку эта переменная изменяется несинхронно с совокупностью других.

3. Да, если использовать схему Горнера.

3.12. ВЫЧИСЛЕНИЕ ИНТЕГРАЛА МЕТОДОМ ТРАПЕЦИЙ

(*условие задачи на стр.* 38)

Вы наверняка заметили, что в формуле, которая приведена в условии задачи, два граничных значения делятся на 2. Поэтому вычислим их отдельно при инициализации S. Умноже-

```
      FUNCTION SOMME (XO,XF,N,FONC)
      DELTAX=(XF-XO)/N
      M=N-1
      X=XO
      S=(FONC(XO)+FONC(XF))/2
      DO 1 I=1,M
      X=X+DELTAX
1     S=S+FONC(X)
      SOMME=S*DELTAX
      RETURN
      END
```

Рис. 78

ние же на Δx отнесем в *самый конец* программы. Теперь нам остается провести вычисления для всех значений x, лежащих между x_0 и x_f, *исключая границы*, т. е. для n — 1 значений. Для этого достаточно одного цикла. Разумеется, отдельно придется вычислить M = N — 1, поскольку выражения не допускаются в качестве параметров DØ-циклов [1]) (см. программу на рис. 78).

[1]) Кстати, в новом стандарте (Фортран 77) это ограничение отсутствует. — *Прим. ред.*

При каждом повторении цикла x увеличивается на Δx и таким образом получаются $x_1, x_2, \ldots, x_{n-1}$, которые и участвуют в вычислении суммы S.

Следующая программа представляется менее корректной:

```
      FUNCTIØN SØMME (X0, XF, N, FØNC)
      DELTAX = (XF - X0)/2
      S = 0
      X = X0
2     A = FØNC (X)
      IF (X .EQ. X0 .ØR. X .EQ. XF) A = A/2
      S = S + A
      IF (X .EQ. XF) GØ TØ 1
      X = X + DELTAX
      GØ TØ 2
1     SØMME = S * DELTAX
      RETURN
      END
```

В самом деле, все, конечно, зависит от значения DELTAX, но может случиться так, что проверка (X.EQ.XF) никогда не даст утвердительного ответа. Если значение DELTAX было вычислено приближенно, то эта ошибка будет сказываться при каждом выполнении присваивания X = X + DELTAX, что может привести к зацикливанию.

ВОПРОСЫ:

1. Почему не написано EXTERNAL FØNC в функции SØMME?
2. Почему написано X = X0 и затем X = X + DELTAX, а не X0 = X0 + DELTAX и FØNC(X0)?

ОТВЕТЫ:

1. Потому что непосредственно за FØNC следует имя переменной, заключенное в скобки, и идентификатор FØNC не описан как имя массива. Этих фактов компилятору Фортрана достаточно, чтобы считать FØNC именем функции.
2. Потому что если в качестве значения X0 передается *константа* (см. условие задачи на стр. 40), то она будет изменена. Существует общее правило, в соответствии с которым подпрограмма никогда не должна изменять значение аргумента, если он рассматривается как *входной* (как *данные*).

В примере на рис. 79 показано применение функции SØMME для вычисления площади круга с радиусом, равным

10. В действительности вычисляется площадь четверти круга, а результат умножается на 4. На рис. 80 приводится таблица полученных значений в зависимости от числа интервалов интегрирования.

```
      EXTERNAL CERCLE
      DO 1 I=50,1000,50
      VAL=SOMME(0.,10.,I,CERCLE)
      S=4*VAL
      PRINT 123, I, S
123   FORMAT (I5,E15.7)
1     CONTINUE
      STOP
      END


      FUNCTION CERCLE (A)
      CERCLE=SQRT(100-A**2)
      RETURN
      END
```

Рис. 79

```
  50  0.3138250E 03
 100  0.3140396E 03
 150  0.3140913E 03
 200  0.3141169E 03
 250  0.3141257E 03
 300  0.3141360E 03
 350  0.3141394E 03
 400  0.3141436E 03
 450  0.3141499E 03
 500  0.3141501E 03
 550  0.3141365E 03
 600  0.3141343E 03
 650  0.3141396E 03
 700  0.3141321E 03
 750  0.3141189E 03
 800  0.3141191E 03
 850  0.3141179E 03
 900  0.3141309E 03
 950  0.3141248E 03
1000  0.3141270E 03
```

Рис. 80

3.13. РЕШЕНИЕ НИЖНЕЙ ТРЕУГОЛЬНОЙ СИСТЕМЫ

(*условие задачи на стр.* 40)

Из рис. 81, на котором воспроизводится программа, следует, что решение не представляет никаких трудностей. Стоит только «запустить» процесс, вычислив предварительно x_1, как в цикле будут найдены все другие значения.

```
      SUBROUTINE RESOL (A,B,X,N,K)
C     РЕШЕНИЕ НИЖНЕЙ ТРЕУГОЛЬНОЙ СИСТЕМЫ
C
      DIMENSION A(N,N), B(N), X(N)
      K=0
      X(1)=B(1)/A(1,1)
      DO 1 I=2,N
      I1=I-1
      C=0
      DO 2 J=1,I1
2     C=C+A(I,J)*X(J)
      IF (C.NE.0) GO TO 1
      K=1
      RETURN
1     X(I)=(B(I)-C)/A(I,I)
      RETURN
      END


      DIMENSION A(5,5), B(5), X(5)
      READ 10, ((A(I,J),J=1,5),I=1,5)
10    FORMAT (5F3.0)
      DATA B/100.,200.,300.,400.,500./
      CALL RESOL (A,B,X,5,KAKA)
      PRINT 1, ((A(I,J),J=1,5),I=1,5),B,X,KAKA
1     FORMAT (7(1X,5F15.4/),/I5)
      STOP
      END
```

```
     5.0000          0.0             0.0             0.0             0.0
     4.0000          8.0000          0.0             0.0             0.0
    -2.0000         -1.0000          6.0000          0.0             0.0
    10.0000          9.0000          1.0000          2.0000          0.0
     8.0000          7.0000          2.0000          5.0000         -1.0000
   100.0000        200.0000        300.0000        400.0000        500.0000
    20.0000         15.0000         59.1667          2.9167       -102.0830

0
```

Рис. 81

4. ВАРИАЦИИ НА ТРИ ТЕМЫ

4.1.1. СОРТИРОВКА ПОСРЕДСТВОМ ПЕРЕСТАНОВКИ СОСЕДНИХ ЧИСЕЛ

(*условие задачи на стр.* 42)

Нетрудно заметить, что в соответствии с этим алгоритмом после каждого просмотра наибольшее число попадает на свое место. При первом просмотре исследуются числа от T(1) до T(1000), при втором — от T(1) до T(999) и т. д.

В программе на рис. 82 обратите внимание на оператор «CALL ALEA», который формирует массив произвольных значений, предназначенных для проверки программы. Поскольку мы оперируем со значениями T(I) и T(I + 1), верхняя гра-

ница N для просмотров вначале равна 999 и затем при каждом просмотре уменьшается на 1.

В цикле DØ1 I одна за другой сравниваются пары чисел (T(I), T(I + 1)). Факт перестановки регистрируется с помощью вспомогательной переменной INDEX, которая первоначально имеет нулевое значение и устанавливается в 1 в случае, когда происходит перестановка в какой-нибудь паре.

```
      INTEGER T(1000)
      T(1)=7283
      CALL ALEA(T,1000)
      N=999
2     INDEX=0
      DO 1 I=1,N
      IF(T(I).LE.T(I+1))GO TO 1
      IGAR=T(I)
      T(I)=T(I+1)
      T(I+1)=IGAR
      INDEX=1
1     CONTINUE
      N=N-1
      IF (INDEX.NE.0) GO TO 2
      PRINT 100,T
100   FORMAT(10I6)
      STOP
      END
```

Рис. 82

ВОПРОС:

Когда переменная INDEX уже равна 1, почему мы упрямо продолжаем при каждой новой перестановке присваивать ей значение 1?

ОТВЕТ:

Два ответа:

1. Если бы мы написали IF (INDEX.EQ.0) INDEX = 1, то делалось бы множество совершенно ненужных проверок.

2. В данном случае значение переменной INDEX всего лишь признак, а отнюдь не счетчик. Поэтому бессмысленно писать INDEX = INDEX + 1.

4.12. СОРТИРОВКА ПОСРЕДСТВОМ ПЕРЕСТАНОВКИ ПО ИНДЕКСАМ

(*условие задачи на стр.* 43)

Опять-таки воспользуемся подпрограммой ALEA и начнем с занесения в массив IT произвольных чисел, а затем напечатаем его, чтобы убедиться, правильно ли работает программа.

В цикле DØ 11 массив IT просматривается от первого до предпоследнего элемента (рис. 83). Внутри этого цикла есть еще один DØ-цикл (по J), в котором I-й элемент массива IT сравнивается со всеми элементами, имеющими индексы от I + 1 до 1000. Каждый раз, когда встречается число IT(J), меньшее IT(I), запоминается его индекс J. Для этого используется вспомогательная переменная M. Вначале без особых на то оснований предполагается, что I-й элемент наименьший и устанавливается M = I, хотя в дальнейшем M, возможно, придется переопределить. В конце первого просмотра M содержит индекс наименьшего элемента IT, и, следовательно, можно поменять местами IT(I) (при первом просмотре I = 1) и IT(M).

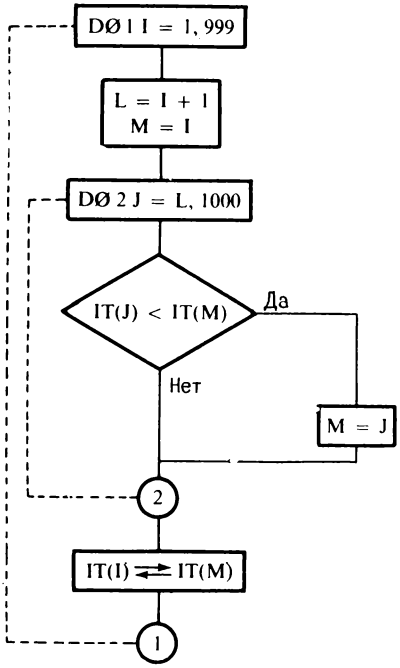


Рис. 83

Теперь та же процедура применяется к части массива IT с индексами от 2 до 1000 (I изменяется лишь до 999, поскольку, очевидно, нет необходимости сравнивать 1000-й элемент с самим собой). В конце второго просмотра будет найдено новое значение M, которое указывает позицию второго по величине элемента, непосредственно большего (или равного) IT(1).

Процесс продолжается до I = 999, т. е. до тех пор, когда массив будет полностью отсортирован. Тогда он печатается, чтобы можно было сопоставить его с исходным массивом.

По сравнению с уже известным методом, при котором перестановки производятся при каждом сравнении, этот алгоритм дает значительный выигрыш времени[1]), так как удается обойтись (да и то не при каждом сравнении) только одним присваиванием вместо трех. Рис. 84 позволяет судить о простоте программы, на что, впрочем, можно было надеяться, только взглянув на блок-схему (рис. 83).

[1]) Обстоятельный анализ метода пузырька и метода простого выбора (названного здесь методом перестановки по индексам) имеется в упомянутой нами книге Д. Кнута. — *Прим. ред.*

```
      DIMENSION IT(1000)
      IT(1)=3579
      CALL ALEA(IT,1000)
      PRINT 100,IT
      DO 1 I=1,999
      L=I+1
      M=I
      DO 2 J=L,1000
      IF(IT(J).LT.IT(M))M=J
2     CONTINUE
      IGARE=IT(I)
      IT(I)=IT(M)
      IT(M)=IGARE
1     CONTINUE
      PRINT 100,IT
100   FORMAT(20I6)
      STOP 1
      END
```

Рис. 84

```
C       СОРТИРОВКА ДВУМЕРНОГО МАССИВА
        SUBROUTINE TRI2DI(IT,I,J,I1,J1,ICLE,IERR)
        DIMENSION IT(I,J),ICLE(J1)
        IERR=1
C  ПРИ ПРОВЕРКЕ АРГУМЕНТА ПРЕДПОЛАГАЕТСЯ, ЧТО ОШИБКА ОДНА
        IF(I.LT.1.OR.I1.LT.1.OR.I1.GT.I) RETURN
        IF(J1.LT.1.OR.J1.GT.J.OR.J.LT.1) RETURN
C  ПРОВЕРКА МАССИВА ICLE
        DO 1 K=1,J1
        L=ICLE(K)
        IF(L.EQ.0) GO TO 2
        IF(L.GT.J.OR.L.LT.1) RETURN
1       CONTINUE
C  ПРИ ЭТОМ ВЫХОДЕ ИЗ ЦИКЛА К НЕ ОПРЕДЕЛЕНО. НУЖНО ЗАДАТЬ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ
        K=J1
        GO TO 51
C  ЭЛЕМЕНТ ICLE НЕ ДОЛЖЕН БЫТЬ НУЛЕВЫМ
2       IF(K.EQ.1) RETURN
        K=K-1
51      CONTINUE
C  АРГУМЕНТЫ ЗАДАНЫ КОРРЕКТНО, МОЖНО НАЧАТЬ СОРТИРОВКУ.
        IERR=0
C  ЕСЛИ СОРТИРУЕТСЯ ТОЛЬКО ОДНА СТРОКА, ТО РАБОТА ЗАКОНЧЕНА,
        IF(I1.EQ.1) RETURN
        I2=I1-1
        M=1
22      II=M
7       KK=II+1
6       IF(KK.GT.I1) GO TO 20
        JC=1
19      CONTINUE
        JJ=ICLE(JC)
        IF(IT(II,JJ).LE.IT(KK,JJ)) GO TO 21
        II=KK
        GO TO 7
21      IF(IT(II,JJ).EQ.IT(KK,JJ)) GO TO 3
4       KK=KK+1
        GO TO 6
3       IF(JC.EQ.K) GO TO 4
        JC=JC+1
        GO TO 19
20      IF(M.GT.I2) RETURN
C  ИНОГДА ПЕРЕСТАНОВКА ЭТ СТОЛБЦОВ НЕ ПРОИЗВОДИТСЯ
        IF(M.EQ.II) GO TO 31
        DO 30 IJ=1,J1
        IPER=IT(II,IJ)
        IT(II,IJ)=IT(M,IJ)
30      IT(M,IJ)=IPER
31      M=M+1
        GO TO 22
        END
```

Рис. 85

4.1.3. СОРТИРОВКА ДВУМЕРНОГО МАССИВА В СООТВЕТСТВИИ С ИЗМЕНЯЕМЫМ КРИТЕРИЕМ

(*условие задачи на стр.* 43)

Многочисленные комментарии, вставленные в программу (рис. 85), объясняют сущность тех проверок, в ходе которых определяется K — число столбцов, используемых в качестве критериев сортировки.

Это упражнение является обобщением предыдущего с добавлением в него проверки правильности аргументов; кроме того, вместо просмотра столбцов массива в естественной последовательности используется порядок, заданный в массиве ICLE.

Блок-схема на рис. 86 поможет провести сопоставления и убедиться, что рассуждения, ведущие к решению, точно такие же, как в предыдущем упражнении.

4.1.4. СОРТИРОВКА ПОСЛЕДОВАТЕЛЬНЫМИ СЛИЯНИЯМИ

(*условие задачи на стр.* 45)

В главной программе (рис. 87) вначале заносятся произвольные значения в массив IT (инструкция CALL ALEA) и вслед за этим многократно вызывается подпрограмма слияния FUS, которая попеременно формирует массив ITT, исходя из IT, а затем IT, исходя из ITT, причем различие в вызове подпрограммы зависит от знака K, который меняется на противоположный всякий раз, когда удваивается длина внутренней последовательности слияния. Заметим, что значение IDEB может стать больше N/2. Подпрограмма FUS внесет необходимые поправки.

Если длина внутренней последовательности больше или равна числу элементов IT, сортировка завершена. Если в этом случае последняя упорядоченная последовательность оказалась в ITT, потребуется дополнительная пересылка без сортировки из ITT в IT. Более подробно операции показаны на блок-схеме рис. 88.

Подпрограмма FUS

Прежде всего надо так определить границы двух первых последовательностей: (I1, N1) и (I2, N2), чтобы исключался выход за пределы массива.

Затем первый элемент первой последовательности сравнивается с первым элементом второй последовательности. Меньший из них пересылается в приемный массив (IT либо ITT в зависимости от случая), и делается переход к следующему элементу в той же последовательности, из которой вы-

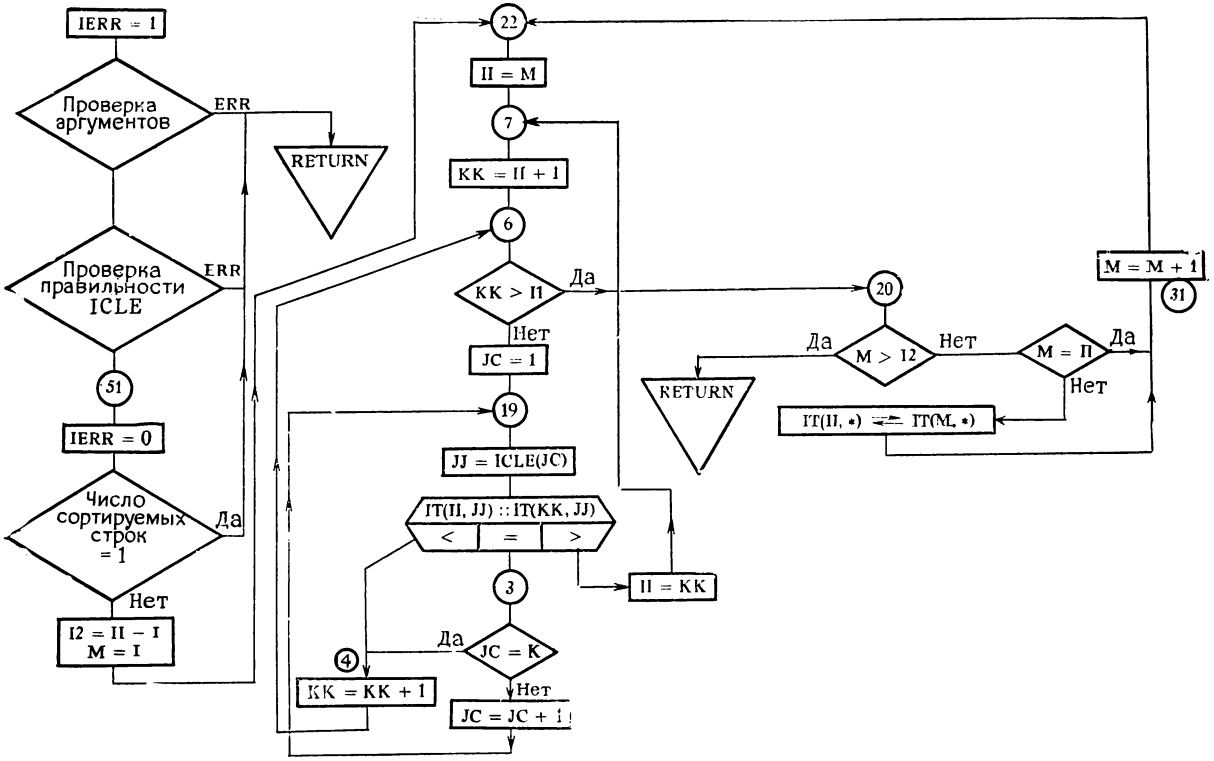

IERR = 1
Проверка аргументов
ERR
RETURN
Проверка правильности ICLE
ERR
51
IERR = 0
Число сортируемых строк = 1
Да
Нет
I2 = II − I
M = I
22
II = M
7
KK = II + 1
6
KK > II
Да
Нет
JC = 1
19
JJ = ICLE(JC)
IT(II, JJ) :: IT(KK, JJ)
<
=
>
II = KK
3
Да
JC = K
4
KK = KK + 1
Нет
JC = JC + 1
20
M > 12
Да
Нет
RETURN
M = II
Да
Нет
IT(II, *) ⇄ IT(M, *)
M = M + 1
31

Рис. 86

```
C       СОРТИРОВКА ПОСЛЕДОВАТЕЛЬНЫМИ СЛИЯНИЯМИ
        DIMENSION IT(10000),ITT(10000)
        N=10000
        K=1
        IDEL=1
        IT(1)=654789
        CALL ALEA(IT,N)
        PRINT 100,IT
20      CALL FUS(IT,ITT,N,IDEL)
        GO TO 30
10      CALL FUS(ITT,IT,N,IDEL)
30      IDEL=IDEL+IDEL
        IF(IDEL.GE.N) GO TO 50
        K=-K
        IF(K) 10,10,20
50      IF (K.NE.1) GO TO 51
        DO 3 I=1,N
3       IT(I)=ITT(I)
51      CONTINUE
        PRINT 101,IT
100     FORMAT(1X,25I5)
101     FORMAT(1H1/(25I5))
        STOP 1
        END


        SUBROUTINE FUS(IT,ITT,N,IDEL)
        DIMENSION IT(1),ITT(1)
        IDEL1=IDEL-1
C
C       УСТАНОВКА ГРАНИЦ СЛИВАЕМЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
C
        I1=1
11      N1=I1+IDEL1
        I2=N1+1
        N2=I2+IDEL1
        IF(N2.LE.N) GO TO 15
        N2=N
        IF(I2.LE.N) GO TO 15
        I2=N
        IF(N1.GT.N) N1=N
15      CONTINUE
C
C       СЛИЯНИЕ 2 ПОСЛЕДОВАТЕЛЬНОСТЕЙ ИЗ IT В ITT
C
        II=I1
        DO 10 K=II,N2
        IF(IT(I1).LE.IT(I2)) GO TO 1
        ITT(K)=IT(I2)
        I2=I2+1
        IF(I2.GT.N2) I2=N1
        GO TO 10
1       ITT(K)=IT(I1)
        I1=I1+1
        IF(I1.GT.N1) I1=N2
10      CONTINUE
        IF(N2.EQ.N) RETURN
        I1=N2+1
        GO TO 11
        END
```

Рис. 87

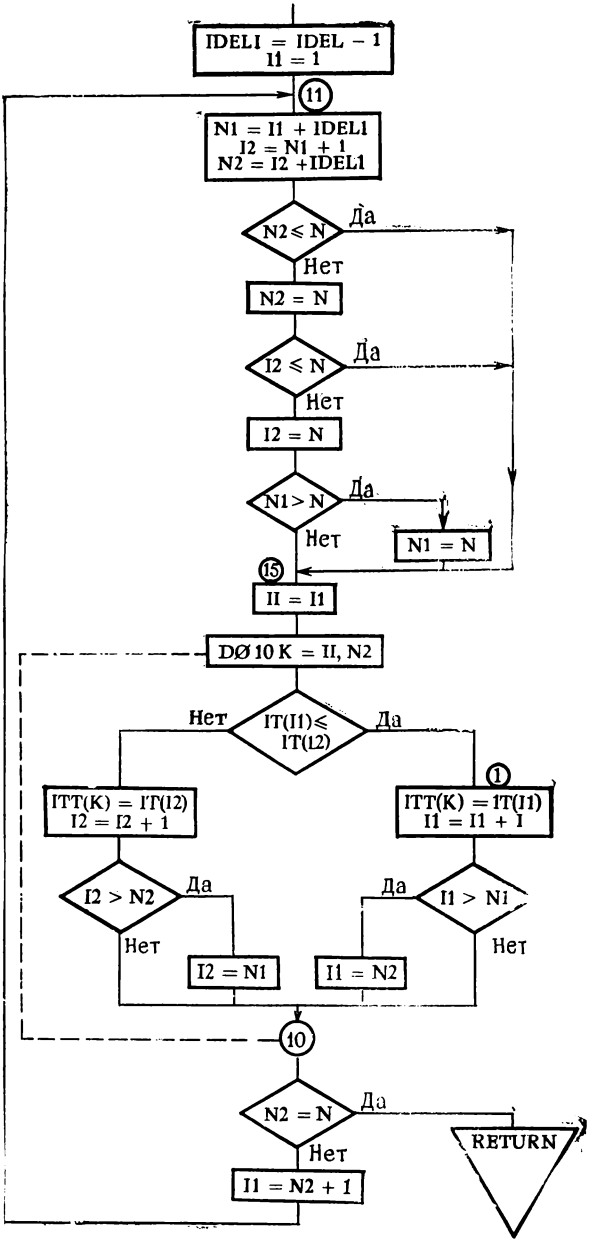

IDELI = IDEL − 1
I1 = 1
11
N1 = I1 + IDELI
I2 = N1 + 1
N2 = I2 + IDELI
N2 ⩽ N
Да
Нет
N2 = N
I2 ⩽ N
Да
Нет
I2 = N
N1 > N
Да
Нет
N1 = N
15
II = I1
DØ 10 K = II, N2
Нет
IT(I1) ⩽ IT(I2)
Да
1
ITT(K) = IT(I2)
I2 = I2 + 1
ITT(K) = IT(I1)
I1 = I1 + 1
I2 > N2
Да
Нет
I2 = N1
I1 > N1
Да
Нет
I1 = N2
10
N2 = N
Да
Нет
I1 = N2 + 1
RETURN

Рис. 88

брали наименьший элемент, чтобы сравнить его с тем же самым элементом другой последовательности, и т. д. до тех пор, пока не будет закончена пересылка всех чисел из обеих последовательностей.

Если теперь N2 меньше N, это означает, что надо продолжить процедуру над новой парой последовательностей, расположенных вслед за теми, которые только что обрабатывались.

Очевидно, N2 не может быть больше N, поскольку значение переменной N2 было откорректировано в самом начале; поэтому, если N2 равно N, то слияние каждого множества последовательностей окончено. В случае когда закончена пересылка одной из последовательностей, операторы IF (I1.GT.N1) и IF(I2.GT.N2) автоматически обеспечивают перепись оставшихся элементов другой последовательности, прибегая к сравнению с известным исходом. В самом деле, последний элемент неисчерпанной последовательности (N1 или N2 в зависимости от случая) заведомо больше или равен самому большому элементу исчерпанной последовательности (N2 или N1 в зависимости от случая). Остается объяснить, почему мы пишем

```
|  | I1 = I1
|  | DØ 10 K = I1, N2
```

Причину нетрудно понять, если вспомнить, что изменять значение индексов начала и конца DØ-цикла внутри этого цикла запрещается. Мы же в цикле DØ пишем

```
или |  | I1 = I1 + 1
    |  | I1 = N2
```

Пользуясь этим искусственным приемом, мы не нарушаем установленных правил. Отметим тем не менее, что большинство компиляторов могли бы корректно обрабатывать упрощенную запись:

```
|  | DØ 10 K = I1, N2
```

Но нельзя быть уверенным, что будет всегда так, поскольку в стандартном Фортране формально такая запись не допускается.

4.1.5. СОРТИРОВКА С ИСПОЛЬЗОВАНИЕМ СТРУКТУРЫ СПИСКА

(*условие задачи на стр.* 46)

LISTE(1) первоначально присваивается значение 0, поскольку еще не введено ни одной карты. В цикле DØ3 та же самая обработка повторяется для каждой из 1000 карт. Если

в колоде более 1000 карт, при помощи обычного завершения DØ-цикла мы проигнорируем лишние карты. Если же в колоде меньше 1000 карт, то цикл завершится благодаря указанию END= в инструкции READ (рис. 89).

LISTE (1) всегда содержит индекс L карты с наименьшим номером. Далее LISTE(L + 1) указывает индекс L′ той карты, номер которой непосредственно больше номера предыдущей карты, затем LISTE (L′ + 1) даст индекс L″ карты, которая следует за ней в порядке возрастания номеров,

```
C       СОРТИРОВКА С ИСПОЛЬЗОВАНИЕМ СТРУКТУРЫ СПИСКА
        INTEGER   Z(19,1000),LISTE(1001)
        LISTE(1)=0
        DO 3K=1,1000
        READ(5,1,END=2)(Z(I,K),I=1,19)
    1   FORMAT(18A4,I8)
        I=LISTE(1)
        J=0
        IF(I.EQ.0)GO TO 11
   12   IF(Z(19,I).GT.Z(19,K))GO TO 11
        J=I
        I=LISTE(I+1)
        IF(I.NE.0)GO TO 12
   11   LISTE(J+1)=K
        LISTE(K+1)=I
    3   CONTINUE
    2   J=0
    4   J=LISTE(J+1)
        IF(J.EQ.0) STOP
C
        PRINT5,(Z(I,J),I=1,19)
    5   FORMAT(1X18A4,I8)
        GO TO 4
        END
```

Рис. 89

и т. д., пока не встретится значение 0, которое указывает на то, что мы нашли карту с наибольшим номером.

Предположим, что мы уже ввели K — 1 карт. Вводим K-ю. Различаем три случая:

A) Эта карта имеет меньший номер, чем те, которые уже считаны.

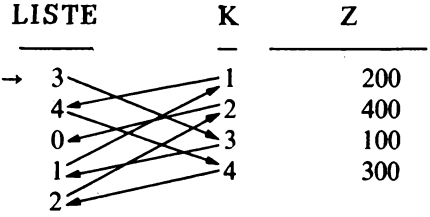
B) — — больший —

C) Номер этой карты заключен между номерами уже считанных карт.

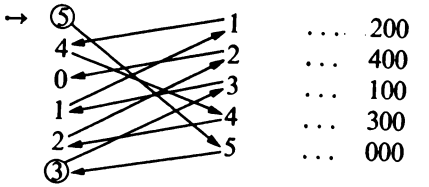
Мы рассмотрим позднее случай ввода первой карты (он выделяется в операторе IF(I.EQ.0)GØTØ 11). А теперь будем сравнивать номер K-й карты с номером уже введенных карт в порядке возрастания номеров, получая индексы введенных карт при помощи массива LISTE.

В случае A, выполнив оператор IF с меткой 12, мы перейдем к оператору с меткой 11. Теперь требуется присвоить

LISTE (1) значение K (поскольку K-я карта имеет наименьший номер), а LISTE (K + 1) присвоить прежнее значение LISTE (1), т. е. I. Например:



Вводим 5-ю карту, которая содержит 000. Теперь необходимо получить



Чтобы это могло осуществиться, надо, чтобы вначале J (роль которого будет объяснена позже) имел значение 0 (оператор с меткой 11: LISTE (J + 1) = K).

Если имеет место случай B, надо найти адрес L карты с самым большим номером. LISTE (L + 1) имеет тогда значение 0. Засылаем K в LISTE (L + 1) и 0 в LISTE (K + 1). К сожалению, мы не знаем индекс карты с самым большим номером и поэтому можем выделить случай A, но не можем различить случаи B и C, пока не сравним K-ю карту со всеми уже введенными.

В ходе этих сравнений мы будем использовать индексы I и J.

I — индекс карты, которая сравнивается с картой K,

J — индекс карты, которая сравнивалась с картой K на предыдущем шаге (т. е. предыдущее значение I).

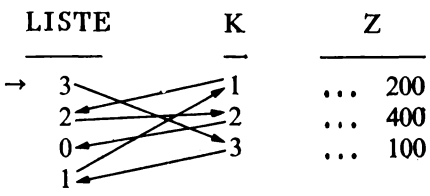
Таким образом, легко понять, почему вначале J присваивается значение 0.

Если новое значение I равно нулю, это означает, что имеет место случай B.

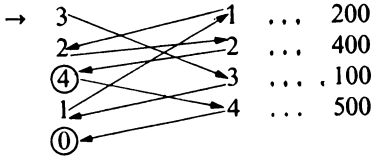
Чтобы перейти к следующей карте, достаточно написать I = LISTE (I + 1).

Оператор IF с меткой 12 даст возможность произвести модификацию указателей (т. е. I и J). Она состоит в том, что K засылается в LISTE (J + 1) (так как J содержит прежнее значение I, которое указывает либо карту с самым большим

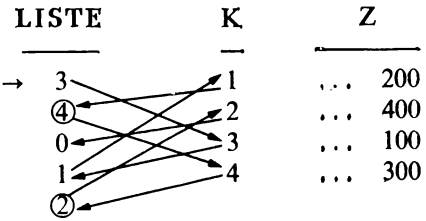
номером (случай B), либо карту с номером, непосредственно большим, чем номер новой карты (случай C)). В случае B I содержит 0, который засылается в LISTE (K + 1), указывая конец последовательности. В противном случае значение I указывает на карту с номером, следующим по величине за номером новой карты, и, таким образом, последовательность установлена. Приведем пример случая B:



Вводим 4-ю карту, которая содержит 500. Мы должны получить



А теперь приведем пример случая C (предполагается, что 4-я карта содержит 300):



Что касается начальных установок, то они сказываются на операторе IF(I.EQ.0) GØTØ 11, который благодаря правильным начальным значениям I и J дает результаты, на которые мы рассчитывали.

Карты перфорируются в последовательности (J = LISTE (J + 1)), до тех пор пока J не станет равным нулю.

Если вводится в точности 1000 карт, то не известно, каково значение J, так как оно зависит от места последней карты среди 999 предыдущих. Вот почему вначале полагают J = 0.

4.2.1. ВЫЧИСЛЕНИЕ π С ПОМОЩЬЮ МЕДЛЕННО СХОДЯЩЕГОСЯ РЯДА

(*см. условие задачи на стр.* 47)

Следует обратить внимание на несколько интересных моментов в решении, которое приводится на рис. 90. Общий член ряда может быть записан в следующем виде:

$$\frac{(-1)^k}{2k+1}.$$

Таким образом, на Фортране можно было бы попытаться написать

$$|\ \ |\ PI = PI + -1 ** K/(2 * K + 1).$$

В этом случае необходимо «—1» заключить в скобки, поскольку два знака операций не могут непосредственно сле-

```
C       ВЫЧИСЛЕНИЕ ПИ
C
6       READ 5, I
5       FORMAT (I7)
        PI=1
        J=2*I+1
        M=-1
        DO 1 K=3,J,2
        PI=PI+M/FLOAT(K)
1       M=-M
        PI=4*PI
        PRINT 200, I, PI
200     FORMAT (6H ПРИ ,I7, 29H ЧЛЕНАХ , ЗНАЧЕНИЕ ПИ РАВНО: ,F12.8)
        GO TO 6
        END
```

Рис. 90

довать друг за другом. Но результат будет неверным, так как берется частное от деления целого числа на целое, а поскольку знаменатель больше 1, результат всегда равен нулю.

Если во избежание этого описать K как вещественную переменную, то произойдет ошибка во время вычисления (—1) ** K, так как при возведении в степень вместо умножений, которые использовались при целом K, будет вычисляться такое выражение:

$$EXP(K * ALØG(-1.))$$

Ясно, что это нас не устроит. Поэтому выберем другое решение:

$$|\ \ |\ PI = PI + (1/K) * M$$

причем K будет изменяться каждый раз на 2, а M будет попеременно принимать значения +1 и —1. Кроме того, не декларируя K как вещественную переменную, можно использовать функцию FLØAT и окончательно записать

$$|\ \ |\ PI = PI + M/FLØAT(K)$$

Переменной M присваивается начальное значение —1, а инструкция M = — M обеспечит желаемую перемену знака (можно было бы также написать M = — 1 * M, что, однако, свидетельствовало бы о крайне поверхностном знании Фортрана). В программе потребовалось предварительное вычисление J = 2 * I + 1, поскольку в цикле K увеличивается с шагом 2. Результаты, представленные на рис. 91, выглядят весьма разочаровывающе.

```
ПРИ      2 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.46666622
ПРИ      3 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  2.89523792
ПРИ      4 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.33968258
ПРИ      5 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  2.97604561
ПРИ      6 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.28373814
ПРИ      7 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.01707172
ПРИ      8 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.25236607
ПРИ      9 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.04183960
ПРИ     10 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.23231506
ПРИ     20 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.18918324
ПРИ     30 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.17384052
ПРИ     40 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.16597652
ПРИ     50 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.16119480
ПРИ     75 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.12842846
ПРИ    100 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.15148354
ПРИ    200 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14654827
ПРИ    300 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14488411
ПРИ    400 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14404488
ПРИ    500 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14353561
ПРИ    750 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14284325
ПРИ   1000 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14248371
ПРИ   2000 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14187336
ПРИ   5000 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14123535
ПРИ   7500 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14089108
ПРИ  30000 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.14057636
ПРИ  50000 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.13602352
ПРИ 100000 ЧЛЕНАХ, ЗНАЧЕНИЕ ПИ РАВНО:  3.13042831
```

Рис. 91

Обратите внимание, что точность, которая весьма медленно возрастала, в конце даже начинает уменьшаться из-за ограниченного числа значащих разрядов в используемой ЭВМ. Это ли не служит явным доказательством того факта, что для увеличения точности результата далеко не достаточно просто увеличить число членов ряда? Требуется еще разобраться в причинах происходящего.

ВОПРОС:

Почему не следует писать PI = PI + FLØAT(M/K)?

ОТВЕТ:

Потому что эта запись ничего не меняет. По-прежнему выполняется целочисленное деление, затем результат

(неверный) преобразуется в вещественную форму. Такая запись обязывает нас предварительно преобразовать в вещественную форму оба операнда перед вычислением.

4.2.2. ВЫЧИСЛЕНИЕ π СТАТИСТИЧЕСКИМ МЕТОДОМ

(*условие задачи на стр.* 47)

Программа на рис. 92 очень простая: выполняются два DØ-цикла; при этом периодически после каждых 10 000 точек печатается найденное значение. Всякий раз, когда получена пара (X, Y), мы проверяем, попадает ли точка в круг единичного радиуса, используя соотношение:

$$X^2 + Y^2 \leqslant 1,$$

потому что по определению $\sqrt{X^2 + Y^2} = R$. Поскольку квадратный корень из числа, меньшего единицы, также меньше единицы, мы избегаем пользоваться операцией SQRT.

Если точка находится внутри круга, K увеличивается на 1, а в конце каждого внутреннего цикла вычисляется и печатается значение π.

Замечания. 1. Подпрограмму HASARD следовало бы записать в виде функции, потому что она дает всего один результат. Но синтаксис Фортрана не позволяет это сделать; функция должна иметь по крайней мере один аргумент. В противном случае компилятор не сможет ее отличить от простой переменной [1]).

2. По результатам мы констатируем, что получены 3 точные цифры, почти 4. Поскольку абсолютная погрешность вычислений имеет порядок $\sqrt{4\pi/500\,000}$, т. е. 0,005, можно считать, что как результат, так и подпрограмма HASARD удовлетворительны.

4.2.3. ВЫЧИСЛЕНИЕ π МЕТОДОМ ВПИСАННЫХ МНОГОУГОЛЬНИКОВ

(*см. условие задачи на стр.* 48)

При вычислении сторон треугольников, изображенных на рис. 29, применяем просто теорему Пифагора. Мы видим, что при обычной точности число значащих цифр не превышает 5 или 6, тогда как при вычислениях с двойной точностью легко получить 15 значащих цифр (рис. 93).

---

[1]) В новом стандрате (Фортран 77) допускаются функции без параметров. Вслед за именем функции пишется пара скобок, например, HASARD( ). — *Прим. ред.*

```
      K=0
      DO 3 J=1,50
      DO 1 I=1,10000
      CALL HASARD (X)
      CALL HASARD (Y)
      IF (X**2+Y**2.LE.1.) K=K+1
1     CONTINUE
      A=10000*J
      B=K/A*4
3     PRINT 2, A, B
2     FORMAT (F13.0,F15.6)
      STOP
      END
```

```
 10000.       3.155600
 20000.       3.162000
 30000.       3.156266
 40000.       3.145700
 50000.       3.143200
 60000.       3.142266
 70000.       3.140971
 80000.       3.141049
 90000.       3.143289
100000.       3.142440
110000.       3.143854
120000.       3.144300
130000.       3.143846
140000.       3.144799
150000.       3.145333
160000.       3.144074
170000.       3.143952
180000.       3.143399
190000.       3.143410
200000.       3.144279
210000.       3.143867
220000.       3.144345
230000.       3.142869
240000.       3.142683
250000.       3.142783
260000.       3.143414
270000.       3.142947
280000.       3.142771
290000.       3.143352
300000.       3.143867
310000.       3.143767
320000.       3.143849
330000.       3.143321
340000.       3.142788
350000.       3.142822
360000.       3.142566
370000.       3.143038
380000.       3.143210
390000.       3.143928
400000.       3.143749
410000.       3.142790
420000.       3.143181
430000.       3.142846
440000.       3.142545
450000.       3.142657
460000.       3.142495
470000.       3.142944
480000.       3.142966
490000.       3.142628
500000.       3.142559
```

Рис. 92

```
C       ВЫЧИСЛЕНИЕ ПИ МЕТОДОМ ВПИСАННЫХ МНОГОУГОЛЬНИКОВ(ПРОСТАЯ ТОЧНОСТЬ)
        N=2
        A=2.
        PII=0
2       C=(A/2.)**2
        B=SQRT(1.-C)
        A=SQRT((1.-B)**2+C)
        PI=A*N
        PRINT 100,PI
100     FORMAT(F15.8)
        IF(PI-PII.LT.1.E-8) GO TO 1
        PII=PI
        N=N+N
        GO TO 2
1       STOP 1
        END
```

```
      2.82842636
      3.06146622
      3.12144279
      3.13654613
      3.14032745
      3.14127350
      3.14150906
      3.14156818
      3.14158058
      3.14158344
      3.14158440
      3.14158440
```

```
C       ВЫЧИСЛЕНИЕ  ПИ МЕТОДОМ ВПИСАННЫХ МНОГОУГОЛЬНИКОВ(ДВОЙНАЯ ТОЧНОСТЬ)
        DOUBLE PRECISION N,A,PI,PII,B,C
        N=2
        A=2
        PII=0
2       C=(A/2.D0)**2
        B=DSQRT(1.D0-C)
        A=DSQRT(( 1.D0-B)**2+C)
        PI=A*N
        PRINT 100,PI
100     FORMAT(D30.20)
        IF(PI-PII.LT.1.D-15) GO TO 1
        PII=PI
        N=N+N
        GO TO 2
1       STOP 1
        END
```

```
      0.28284271247461900000D 01
      0.30614674589207180000D 01
      0.31214451522580520000D 01
      0.31365484905459390000D 01
      0.31403311569547530000D 01
      0.31412772509327730000D 01
      0.31415138011443010000D 01
      0.31415729403670910000D 01
      0.31415877252771590000D 01
      0.31415914215111990000D 01
      0.31415923455701170000D 01
      0.31415925765848720000D 01
      0.31415926343385620000D 01
      0.31415926487769840000D 01
      0.31415926523865900000D 01
      0.31415926532889910000D 01
      0.31415926535145910000D 01
      0.31415926535709910000D 01
      0.31415926535850510000D 01
      0.31415926535886160000D 01
      0.31415926535894970000D 01
      0.31415926535897170000D 01
      0.31415926535897720000D 01
      0.31415926535897860000D 01
      0.31415926535897890000D 01
      0.31415926535897900000D 01
```

Рис. 93

4.2.4. ВЫЧИСЛЕНИЕ π ПО ФОРМУЛЕ МЕЧИНА

(*см. условие задачи на стр.* 49)

Самой трудной проблемой оказывается определение Arc tg 1/239, так как надо вычислить

$$\frac{1}{239} - \frac{1}{3 \times 239^3} + \frac{1}{5 \times 239^5} \cdots$$

Действительно, уже во втором знаменателе получим

$$239^3 = 13651919.$$

Нет смысла продолжать, по крайней мере ведя вычисления в целых числах. Если мы хотим получить какой-либо толк от вычислений, надо использовать двойную точность, т. е. плавающую запятую. Программа на рис. 94 дает удовлетворительные результаты (рис. 95). Чтобы присвоить начальные значения U и V, очень важно написать D0 после 5. и 239. для полной уверенности в том, что вычисления будут проводиться с двойной точностью. В самом деле, проведение последующих вычислений с большим количеством значащих цифр ничего бы не дало, если бы V имело лишь ограниченное число значащих цифр. Если бы мы написали V = 1/239, то, очевидно, получили бы 0. Если бы мы поставили десятичную точку только в одну из констант, мы бы получили в качестве частного вещественное число с обычной точностью, которое было бы «дополнено» справа нулями при преобразовании в двойную точность.

Затем вычисляется по отдельности каждый член ряда, чтобы можно было напечатать последовательные значения, принимаемые U и V, и таким образом проследить за изменением точности.

Необходимо, чтобы *все* переменные были перечислены в операторе DØUBLE PRECISIØN, так как окончательную точность определяет самое слабое звено в цепи вычислений.

При изучении результата можно констатировать, что, как мы и предвидели, V стабилизируется на третьем шаге (при A = 7). Отсюда можно заключить, что для получения достаточного числа значащих цифр совершенно бесполезно брать более 8 членов (учитывая разницу в порядках U и V). Но если бы при этих же 8 членах мы могли выполнить вычисления с четырехкратной точностью (FORTRAN EXTENDED IBM) либо с двойной точностью (CONTROL DATA), мы бы получили выигрыш в значащих цифрах.

В заключение следует заметить, что, как показали эти «вариации на тему π», никогда не надо необдуманно увели-

```
C       ВЫЧИСЛЕНИЕ ПИ ПО ФОРМУЛЕ МЕЧИНА
C
        DOUBLE PRECISION A,B,U,V,PI
        U=1/5.D0
        V=1/239.D0
        B=-1
        A=3
1       U=U+B/(A*5**A)
        V=V+B/(A*239**A)
        PI=4*(4*U-V)
        PRINT 2, A,U,V,PI
2       FORMAT (D10.2,2D32.24,D24.16)
        B=-B
        A=A+2
        IF (A.LE.15.D0) GO TO 1
        STOP
        END
```

Рис. 94

```
        DIMENSION K(17)
        DO 33 N=2,51
        M=1000000
        DO 1 I=1,17
        K(I)=M/N
        M=MOD(M,N)*1000000
1       CONTINUE
        PRINT 3, N, K
33      CONTINUE
4       STOP
3       FORMAT (I5,5X,2H0.,17I6)
        END
```

Рис. 96

```
0.30D 01  0.197333333333333300000000D 00  0.418407600181825300000000D-02  0.3140597029326060D 01
0.50D 01  0.197397333333333300000000D 00  0.418407600207472500000000D-02  0.3141621029325034D 01
0.70D 01  0.197395504761904700000000D 00  0.418407600207472200000000D-02  0.3141591772182177D 01
0.90D 01  0.197395561650793600000000D 00  0.418407600207472200000000D-02  0.3141592682404399D 01
0.11D 02  0.197395559788975400000000D 00  0.418407600207472200000000D-02  0.3141592652615308D 01
0.13D 02  0.197395559851990800000000D 00  0.418407600207472200000000D-02  0.3141592653623554D 01
0.15D 02  0.197395559849806300000000D 00  0.418407600207472200000000D-02  0.3141592653588601D 01
```

Рис. 95

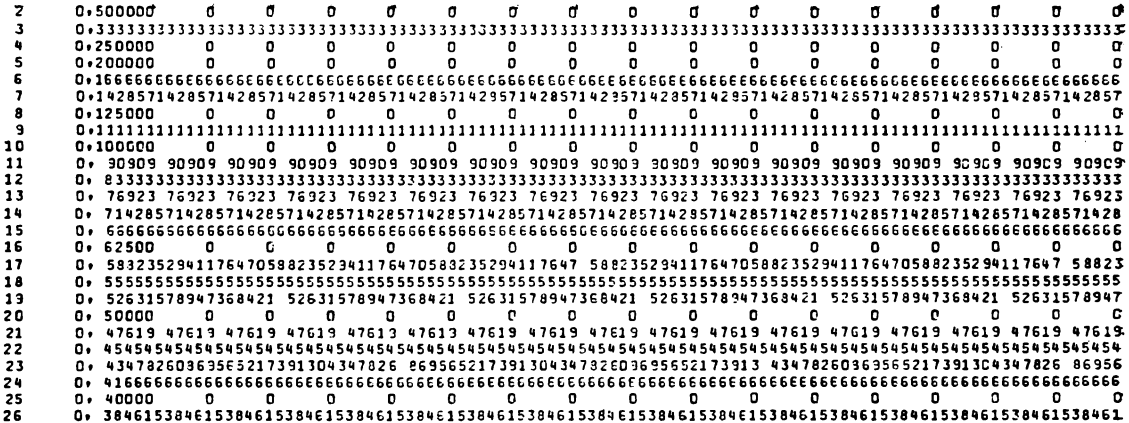


Рис. 97

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	720
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5040
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40320
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	362880
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3628800
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39916800
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	479 1600
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6227 20800
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87178291200
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1307674368000	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20922789888000	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	355687428 96000	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6402373705728000	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121645102408832000		
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2432902 8176640000		
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	51 90942171709440000		
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1124 727777607680000		
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25852 16738884976640000		
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	620448401733239439360000		
25	0	0	0	0	0	0	0	0	0	0	0	0	0	15511210 43330985984			0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	403291461126605635584			0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	10888869450418352160768			0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	304888344611713860501504			0
29	0	0	0	0	0	0	0	0	0	0	0	0	8841761993739701954543616				0
30	0	0	0	0	0	0	0	0	0	0	0	0	265252859812191 58636308480				0
31	0	0	0	0	0	0	0	0	0	0	0	0	8222838654177922817725562880				0
32	0	0	0	0	0	0	0	0	0	0	0	0	263130836933693530167218 12160				0
33	0	0	0	0	0	0	0	0	0	0	0	0	8683317618811886495518194401280				0
34	0	0	0	0	0	0	0	0	0	0	0	295232799 39604140847618609643520					0
35	0	0	0	0	0	0	0	0	0	0	0	10333147966386144929666651337523200					0

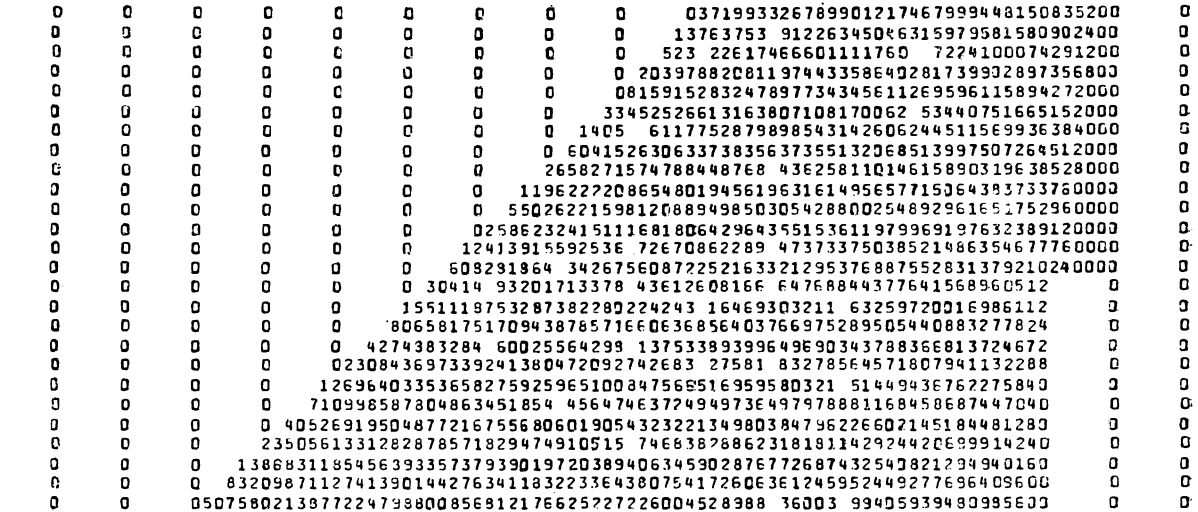


Рис. 99

чивать число членов ряда, используемых в вычислениях, в целях улучшения результата. Порой достаточно иметь немного членов, лишь бы каждый из них имел достаточное число значащих цифр.

4.3.1. ОБРАТНЫЕ 50 ПЕРВЫХ ЦЕЛЫХ ЧИСЕЛ

(*см. условие задачи на стр.* 50)

Решение, представленное на рис. 96, не требует комментариев! (Результаты, полученные с помощью этой программы, см. на рис. 97.)

4.3.2. n ФАКТОРИАЛ

(*см. условие задачи на стр.* 50)

Переменная M, которой вначале присваивается значение 0, содержит перенос в следующее слово. По поводу программы,

```
      DIMENSION K(17)
      DATA K/17*0/
      K(1)=1
      DO 2 I=1,70
      M=0
      DO 3 J=1,17
      K(J)=K(J)*I+M
      M=K(J)/1000000
      K(J)=MOD(K(J),1000000)
3     CONTINUE
2     PRINT 4,I,K(17),K(16),K(15),K(14),K(13),K(12),K(11),K(10),K(9),
     1  K(8),K(7),K(6),K(5),K(4),K(3),K(2),K(1)
4     FORMAT (I6,10X,17I6)
      STOP
      END
```

Рис. 98

изображенной на рис. 98, стоит, по-видимому, высказать несколько замечаний:

- чтобы установить существование переноса и его величину, достаточно иметь частное. Остаток от деления при каждом частичном вычислении может иметь максимум 6 цифр (результат см. на рис. 99),
- для печати мы вынуждены указать индексы всех переменных в списке, так как в неявном цикле DØ нельзя задать отрицательный шаг (за исключением особо «терпимых» компиляторов).

Массив результатов представлен на рис. 99, который из полиграфических соображений разделен на две части, стр. 136 и 137.

4.3.3. ЧИСЛО e, ОСНОВАНИЕ НАТУРАЛЬНЫХ ЛОГАРИФМОВ

(*см. условие задачи на стр.* 51)

Решение приведено на рис. 100. Необходимо иметь три переменных (или, точнее, три массива): J, K и L. Чтобы получить 1/n!, мы будем делить 1/(n — 1)! на n (ср. с решением задачи 3.10, стр. 111).

```
      DIMENSION J(17), K(17), L(17)
      DATA J,K/17*0, 17*0/
      K(1)=500000
      J(1)=K(1)
      DO 3 N=3,70
      M=0
      DO 2 I=1,17
      L(I)=(K(I)+M)/N
      M=MOD(K(I)+M,N)*100000
2     CONTINUE
      DO 5 I=1,17
5     K(I)=L(I)
      M=0
      I=17
8     CONTINUE
      J(I)=J(I)+L(I)+M
      M=0
      IF (J(I).LT.1000000) GO TO 7
      J(I)=J(I)-1000000
      M=1
7     CONTINUE
      I=I-1
      IF (I.GE.1) GO TO 8
      PRINT 4, N, J
4     FORMAT (I4,6X,2H2.,17I6)
3     CONTINUE
      STOP
      END
```

Рис. 100

K будет содержать 1/(n — 1)!,

L будет содержать частное от деления K на n,

J накапливает сумму последовательных значений 1/n!: это и будет значение e.

K присваивается начальное значение 5000...0, что представляет собой 1/2!. (Если бы мы начали с 1/1!, частное от деления дало бы 7 цифр, что вызвало бы некоторые затруднения.) По тем же причинам вычисляется лишь *дробная* часть числа e (что строго согласуется с условием задачи), а «2» заранее включена в формат печати.

Получение двух последовательных значений (J = L + J) производится очень просто, начиная справа (самое большое значение индекса). Так как в этом точном случае перенос может принимать лишь значение 0 или 1, желательно избе-

3	2,666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
4	2,708333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333332
5	2,716666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666665
6	2,718055555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555553
7	2,718253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968253968251
8	2,718278769841269841269841269841269841269841269841269841269841269841269841269841269841269841269841269838
9	2,718281525573192239858906525573192239858906525573192239858906525573192239858906525573192239858906525559
10	2,718281801146384479717813 51146384479717813 51146384479717813 51146384479717813 51146384479717813 51142
11	2,718281826198492865159531826198492865159531826198492865159531826198492865159531826198492865159531826194
12	2,718281828286168563946341724119501897279675 57452835230613008390786168563946341724119501897279675 57448
13	2,718281828446759002314557870113425668981224536790 92335647891203446759002314557870113425668981224536775
14	2,718281828458229747912287594827277366959906642446324986 7525690065372605 55144737684420224102763785298
15	2,718281828458994464285469576474867480158485449490740496 31501322506613511904517195522486527777533 68532
16	2,718281828459 42259 58793450327641862233396624931 164654079959799534191068582602974137365671757206148734
17	2,718281828459 450735160477958486050611789796 35251 32698900735 4065225042504843314055887974344245741687
18	2,718281828459 45226708117481710869683342623135824366934 94775848761393596611634444051361435599081274628
19	2,718281828459 45234928752728335199400298604372696647683315514840587507731 38307661419544249349335776351
20	2,718281828459 45235339734490666415886146403434540261720776551790178813437759641322287953390 36848501447
21	2,718281828459 45235359357431729807147577251 8913767151131839263968875614270181 20424544301498159631213
22	2,718281828459 45235360247110869 52204705925898658 17397966170512777514804111569188521662070200945455293
23	2,718281828459 45235360285792570758511546303 67777332626 39402306203977377582933891482406321 14110 99818
24	2,718281828459 45235360287404308329607664652116490637427261203630930 79984810907420772437331464658626673
25	2,718281828459 45235360287469777632451509386 78439169619308075633919124089100026361944 385718826803567747
26	2,718281828459 45235360287471257428714734183538514113165156032301341779631572684782758330927283373719326
27	2,718281828459 45235360287471349265613372138999998370333520771435320396503516116576121823236742658650865
28	2,718281828459 45235360287471352545502609208837908522375248083547248204248942667711599 90819223347398419
29	2,718281828459 45235360287471352658602238073315077837962893852930418128653957376371443134528964060803507
30	2,718281828459 45235360287471352662372225702130983481815815378576523792800791199993437935985955417917009
31	2,718281828459 4523536028747135266249383820628633527677881284714575 3 7773269710432857123129729332662505
32	2,718281828459 45235360287471352662497638597 41190 20371406518 38541420741159663884088972727972267498404
33	2,718281828459 45235360287471352662497753760397397739874212386853474402952307844291702 59079434174614643
34	2,718281828459 45235360287471352662497757147554933261036 59618289207725958518084891925973383888936588646
35	2,718281828459 45235360287471352662497757244330862847354969539187371535187266948909075228078301929787903
36	2,718281828459 45235360287471352662497757247 19083113641605925878987196554732195131773818486480068487882

37	2,718281828459	45235360287471352662497757247	9173771543311366639 3281464686196098557022513174075515331761 1
38	2,718281828459	45235360287471352662497757247	9364967863817692095791536923846700637123 7 840534133976048
39	2,718281828459	45235360287471352662497757247	9369873333574205639199231083786459664818 6 308839204727462 13
40	2,718281828459	45235360287471352662497757247	9369992895318118477774173437784953640510975644568495 1 70855
41	2,718281828459	45235360287471352662497757247	9369995884628945620178684226 9 6868127722984731324993834706 5
42	2,718281828459	45235360287471352662497757247	93699959558 3012933093077340933580377418 5 875719967237584 03
43	2,718281828459	45235360287471352662497757247	9369995957458223 8 8352725296318760111323581531502462954015
44	2,718281828459	45235360287471352662497757247	9369995957495842229647595147556837275534945639421850248118 8
45	2,718281828459	45235360287471352662497757247	9369995957496678194732313427955169748077225361339997 0 26236
46	2,718281828459	45235360287471352662497757247	9369995957496696367886329042746437845306 4 5355293717584243 2
47	2,718281828459	45235360287471352662497757247	9369995957496696754549180438 8 582652822617514239913717108 61
48	2,718281828459	45235360287471352662497757247	9369995957496696762E0465E5095570637924536703462971667456119
49	2,718281828459	45235360287471352662497757247	9369995957496696762769D5398038872169580525188107059592080 53
50	2,718281828459	45235360287471352662497757247	9369995957496696762772341929 8 535485387228351176606 45 43061
51	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 63994 1759425599088053544407 0255511
52	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 76392 1690282747680448578605920 1904
53	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766259414 18 835425766357925118235 34
54	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766302733535512474983422741127980 08
55	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303521 16 1429667373693741463390 7
56	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535224862851128386842116452405
57	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535471618082239947351813965 89
58	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475872385187465954066542 47
59	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475944492017084913426755 53
60	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945693797578562749425 84
61	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713498898130771108 95
62	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713816661349 61 16835
63	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821705210226661 35
64	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821784020548794 05
65	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821785233015208 39
66	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821785251385992 84
67	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821785251665 18245
68	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821785251664214 65
69	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821785251664273 08
70	2,718281828459	45235360287471352662497757247	9369995957496696762772 4 766303535475945713821785251664273 91

Рис. 10i

жать деления и вычисления остатка и ограничиться операциями сравнения и вычитания. Результаты можно найти на рис. 101, стр. 140 и 141.

5. ОБРАБОТКА ТЕКСТОВ

5.2. СЕМЕЙСТВО CARCAR

(*условие задачи на стр.* 54)

Подпрограмма APLCAR

Чтобы выделить литеру в массиве, зная ее номер, I, надо определить

K — индекс слова, в котором она находится,
ее место в этом слове.

```
SUBROUTINE APLCAR(IZ,I,MOT)
DIMENSION IZ(1),MASQUE(4)
LOGICAL MASQUE,L,BLANC
EQUIVALENCE (K,L)
DATA MASQUE/ZFF000000,ZFF0000,ZFF00,ZFF/,BLANC/Z40404000/
K=(I-1)/4+1
K=IZ(K)
J=MOD(I-1,4)
L=L.AND.MASQUE(J+1)
K=K/256**(3-J)
L=L.AND.MASQUE(4).OR.BLANC
MOT=K
RETURN
END
```

Рис. 102

Если в слове 4 литеры, то, очевидно,

$$K = (I - 1)/4 + 1.$$

Так как впоследствии нам не понадобится K, туда тотчас же засылается слово массива, содержащее искомую литеру (K = IZ(K)).

Если предположить, что в этом слове 4 литеры, перенумерованные слева направо 0, 1, 2 и 3, то номер J искомой литеры определяется следующим образом (рис. 102):

$$J = \text{MØD}\,(I - 1,4).$$

Теперь надо стереть литеры, находящиеся справа и слева от той, которая нам нужна. В действительности необходимо избавиться лишь от тех, что находятся справа, по причине внутреннего представления отрицательных переменных в допол-

нительном коде, поскольку в Фортране сдвиги можно реализовать лишь с помощью деления.

Чтобы выделить литеры, мы используем 4 «маски», составленные следующим образом (во внутреннем двоичном представлении):

```
MASQUE (1)   11111111 00000000 00000000 00000000
MASQUE (2)   00000000 11111111 00000000 00000000
MASQUE (3)   00000000 00000000 11111111 00000000
MASQUE (4)   00000000 00000000 00000000 11111111
```

Таким образом, номер выбранной маски есть J + 1 (так как индексы в Фортране начинаются с 1, а не с 0).

Логическая операция .AND. (определенная над машинными словами) выполняет логическое умножение K и выбранной маски. Есть, правда, одна трудность — эта операция применима лишь к двум величинам типа LØGICAL. Поэтому приходится использовать оператор EQUIVALENCE.

Чтобы определить маску, используется оператор DATA, в котором двоичные величины представлены шестнадцатеричными константами (типа Z), если рассматривается вычислительная машина фирмы IBM, либо восьмеричными (типа Ø) для машин UNIVAC или CONTROL DATA.

Например,

```
MASQUE (1)   FF 00 00 00     (F16 = 1111₂)
MASQUE (2)   00 FF 00 00
MASQUE (3)   00 00 FF 00
MASQUE (4)   00 00 00 FF
```

Промежуточная переменная K типа INTEGER занимает ту же ячейку памяти, что и переменные L типа LØGICAL. Поэтому записывая

$$L = L.AND.MASQUE(J + 1),$$

получаем искомое значение переменной K. Теперь надо поместить эту литеру в правую часть K. Этот сдвиг будет осуществлен делением K на степень 256 (0, 1, 2 или 3 в соответствии со значением J, которое равно 3, 2, 1 или 0), так как каждая литера занимает 8 битов ($2^8 = 256$). Здесь уже надо использовать имя «K», так как нельзя выполнять арифметические операции над логическими переменными, но безразлично, идет ли речь о K или L, вспомним, что в памяти это одно и то же!

После деления искомая литера, в самом деле, находится в правой части K. Однако если вначале она была крайней слева (J = 0) и к тому же «отрицательной» (самый левый

бит равен 1), то теперь слева от нее будут 24 бита «1» (распространение знака). Чтобы избавиться от этих возможных неуместных единиц, достаточно применить еще одно логическое умножение на MASQUE(4) (00 00 00 FF).

И наконец, необходимо еще добавить три пробела слева. Для этого достаточно применить логическое сложение (.ØR.). с маской BLANC, которая содержит три пробела слева и восемь двоичных нулей справа.

Теперь можно заслать K в MØT: это искомый результат. Нам удалось ввести промежуточную переменную K, поскольку мы пользовались оператором EQUIVALENCE (K, L). В самом деле, известно, что нельзя применять оператор EQUIVALENCE к формальным параметрам подпрограммы.

Вернемся к примеру, который обсуждался в условии задачи, и посмотрим, что произойдет, если написать

CALL APLCAR(K(1), 13, MØT)

(содержимое слов памяти представлено в шестнадцатеричном коде)

K	00	00	00	04		
K	D4	D5	D6	D7		литеры „MNØP“
J	00	00	00	00		
L	D4	00	00	00		
K	FF	FF	FF	D4		
L	00	00	00	D4	после .AND.	
	40	40	40	D4	после .ØR.	литеры „∧∧∧M“

Подпрограмма SØRCAR

Вначале найдем машинное слово, содержащее литеру, которую надо заменить, и зашлем это слово в K (рис. 103).

```
SUBROUTINE SORCAR(IZ,I,MOT)
DIMENSION IZ(1),MASQUE(4)
LOGICAL MASK,MASQUE,L,MM
EQUIVALENCE (K,L),(M,MM)
DATA MASQUE/Z00FFFFFF,ZFF00FFFF,ZFFFF00FF,ZFFFFFF00/
DATA MASK/ZFF/
J=MOD(I-1,4)+1
K1=(I-1)/4+1
K=IZ(K1)
M=MOT
MM=MM.AND.MASK
M=M*256**(4-J)
L=L.AND.MASQUE(J).OR.MM
IZ(K1)=K
RETURN
END
```

Рис. 103

Затем будем освобождать, используя логическое умножение на MASK (00 00 00 FF), все то, что может содержаться в трех левых позициях слова MØT. Тогда единственной крайней справа литере будут предшествовать двоичные нули. Поскольку нельзя изменять переменную MØT (она передает данные из вызывающей программы), мы пересылаем ее значение в переменную M, имеющую тот же тип (INTEGER). Логическое умножение (.AND.) применяется к MM, описанной как LØGICAL, но отождествленной с M в инструкции EQUIVALENCE.

Теперь можно переместить эту единственную литеру в ту же позицию, что и заменяемая литера в K. С помощью умножения на подходящую степень 256 имитируется сдвиг влево. Здесь используется переменная M, так как выполняется арифметическая операция.

Затем занесем 0 в ту же позицию в K с помощью другого логического умножения на один из элементов маски и поместим туда литеру из MM, используя логическое сложение.

```
MASQUE(1)  00 FF FF FF
MASQUE(2)  FF 00 FF FF   заметим, что массив
MASQUE(3)  FF FF 00 FF   MASQUE совсем не
MASQUE(4)  FF FF FF 00   тот, который применялся
                         в подпрограмме APLCAR
```

Результат присваивается переменной L (LØGICAL) и оттуда под именем K (INTEGER) пересылается в IZ.

Предположим, что написаны следующие инструкции:

```
      DATA W/4HTUVW/
      CALL SØRCAR(K(1), 7, W)
```

Проследим (инструкция за инструкцией), как изменяется содержимое памяти:

```
I     00 00 00 07
MØT   E3 E4 E5 E6              литеры "TUVW"
J     00 00 00 03
K1    00 00 00 02
K     C5 C6 C7 C8              литеры "EFGH"
M     E3 E4 E5 E6              литеры "TUVW"
MM    00 00 00 E6
M     00 00 E6 00
L     C5 C6 00 C8  после .AND.
      C5 C6 E6 C8  после .ØR.  литеры "EFWH"
```

Остается только снова заслать L т. е. K, в IK(K1), чтобы закончить работу.

Пример

Небольшая программа (рис. 104) вводит карту и буквально «переворачивает» ее содержимое, засылая литеру, которая

```
      INTEGER CARTE(20), RESUL(20)
      READ 1, CARTE
1     FORMAT (20A4)
      DO 2 I=1,80
      CALL APLCAR (CARTE(1), I, LETTRE)
2     CALL SORCAR (RESUL(1), 81-I, LETTRE)
      PRINT 3, CARTE, RESUL
3     FORMAT (1X, 20A4)
      STOP
      END
```

Рис. 104

была в столбце 1 в столбец 80, литеру из 2-го столбца в 79 и т. д.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ. VOICI UN PALINDROME:' ELU PAR CETTE CRAPULE' *******
******* 'ELUPARC ETTEC RAP ULE ':EMORDNILAP NU ICIOV .ZYXWVUTSRQPONMLKJIHGFEDCB
```

5.3.1. PL/1 НА ФОРТРАНЕ. ФУНКЦИЯ INDEX

(*условие задачи на стр.* 56)

На рис. 105 мы видим, что IFIN определяет позицию, в которой заканчивается просмотр цепочки IT, а N — позицию,

```
C        НЕКОТОРЫЕ СРЕДСТВА  PL1 НА ФОРТРАНЕ.
      FUNCTION INDEX(IT,L,N,ICHCAR,M)
      LOGICAL*1 IT(1),ICHCAR(1)
      IFIN=N+L-M
      DO 1 I=N,IFIN
      DO 2 J=1,M
      K=I+J-1
      IF(IT(K).NE.ICHCAR(J))GO TO 1
    2 CONTINUE
      INDEX= I-L+1
      RETURN
    1 CONTINUE
      INDEX =0
      RETURN
      END
```

Рис. 105

с которой просмотр начинается. Будем сравнивать каждую подцепочку из M литер, начинающуюся в этих пределах, с M литерами ICHCAR; это делается в цикле DØ2. Хотя сравниваемые переменные имеют тип LØGICAL, их тем не менее можно сравнивать между собой при условии, что используются лишь операции .NE. или .EQ. Сравнение производится над 8 битами каждого байта (литера) и, следовательно, в этом

случае оно правомерно. Если цикл DØ2 заканчивается нормально, это означает, что проверка дала ответ «РАВНО» во всех случаях. Теперь достаточно вычислить INDEX.

Напротив, нормальное завершение цикла DØ1 свидетельствует о том, что искомой конфигурации не существует, поскольку внутренний DØ-цикл ни разу не закончился нормально. Это оправдывает занесение в INDEX значения 0.

5.3.2. PL/1 НА ФОРТРАНЕ. ПОДПРОГРАММА SUBSTR

(*условие задачи на стр.* 56)

Это решение (рис. 106) настолько простое, что не требует комментариев: входная цепочка по одной литере пересылается в выходную, начиная с указанного места.

```
      SUBROUTINE SUBSTR(IT,N,L,ITT,NN)
      LOGICAL*1 IT(1),ITT(1)
      K=0
      M=L+N+1
      DO 1I=N,M
      ITT(NN+K)=IT(I)
      K=K+1
    1 CONTINUE
      RETURN
      END
```

Рис. 106

5.3.3. PL/1 НА ФОРТРАНЕ. ПОДПРОГРАММА TRANSF

(*условие задачи на стр.* 57)

Вначале вычисляется NLIT, номер последней литеры, которая будет обрабатываться. Затем во внешнем цикле DØ1I

```
      SUBROUTINE TRANSF(IT,N,LIT,IREM,IDEC,L)
      LOGICAL*1 IT(1),IREM(1),IDEC(1)
      NLIT=N+LIT-1
      DO 1 I=N,NLIT
      IC=IT(I)
      DO 2J=1,L
      IF(IC.NE.IDEC(J)) GO TO 2
      IT(I)=IREM(J)
      GO TO 1
    2 CONTINUE
    1 CONTINUE
      RETURN
      END
```

Рис. 107

просматривается каждая из литер, а во внутреннем цикле DØ2J они последовательно сравниваются с каждой из L литер цепочки IDEC (см. рис. 107).

Когда при сравнении обнаруживается совпадение, то эта литера в массиве IT заменяется на литеру из IREM, имеющую тот же номер, что и соответствующая литера в IDEC.

5.3.4. PL/1 НА ФОРТРАНЕ. ИНВЕРТИРОВАНИЕ ЦЕПОЧКИ

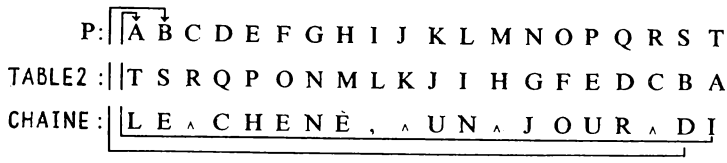
(*условие задачи на стр.* 57)

Эта задача хорошо известна тем программистам, которые пишут на PL/1 и на языке ассемблера 360/370 (функция

```
C
C
C       ИНВЕРСИЯ АЛФАВИТНО -ЦИФРОВОЙ ЦЕПОЧКИ
C
        LOGICAL*1 CHAINE(20), TABLE1(20), TABLE2(20), P(20)
        DATA TABLE1/'A','B','C','D','E','F','G','H','I','J','K','L',
     1             'M','N','O','P','Q','R','S','T'/,
     2   TABLE2/'T','S','R','Q','P','O','N','M','L','K','J','I','H',
     3          'G','F','E','D','C','B','A'/
     4  READ 1, CHAINE
     1  FORMAT (20A1)
        DO 2 I=1,20
     2  P(I)=TABLE1(I)
        CALL TRANSF (P, 1, 20, CHAINE, TABLE2, 20)
        PRINT 3, CHAINE, P
     3  FORMAT (1X 20A1)
        GO TO 4
        END
```

Рис. 108

TRANSLATE или команда TR). Этапы выполнения этой операции иллюстрируются следующей схемой:



Посмотрим, что происходит с двумя первыми литерами: мы модифицируем цепочку P, заменяя в ней 1-ю литеру J-й литерой из цепочки CHAINE, при этом J определяется следующим образом: TABLE2(J) = P(I) (рис. 108). Поскольку TABLE2 и P симметричные (взаимно инверсные) цепочки, каждый раз, когда мы хотим заменить литеру в цепочке P, соответствующая литера в TABLE2 будет симметрична, и для замены нам надо взять литеру именно в этой позиции, но из цепочки CHAINE; в результате произойдет замена P (дуб-

ликат TABLE1) цепочкой, инверсной по отношению к CHAINE.

Необходимо использовать вспомогательный массив P, иначе при выполнении операции была бы уничтожена цепочка TABLE1.

5.4. МЕРСИ, Г. ДОРНБУШ

(*условие задачи на стр.* 57)

Сначала ищем начальную букву слова MERCI: «M». Как только она найдена (рис. 109), в DØ-цикле проверяется, соответствует ли этому слову последующие буквы в тексте: вторая «E», третья «R» и т. д. Если выход из цикла произошел

```
      DIMENSION MERCI(240),M(5)
      DATA M/1HM, 1HE, 1HR, 1HC, 1HI/
      READ 1, MERCI
1     FORMAT (80A1)
      N=0
      I=1
5     IF (MERCI(I) .NE. M(1)) GO TO 4
      J=1
      DO 2 K=1,4
      IF (MERCI(I+K) .NE. M(K+J)) GO TO 3
2     CONTINUE
      N=N+1
      K=5
3     I=I+K
4     I=I+1
      IF (I .LE. 236) GO TO 5
      PRINT 6, N
6     FORMAT (7H IL Y A  , I3, 7H MERCIS )
      STOP
      END
```

Рис. 109

из-за неравенства (IF), это означает, что мы попали не на слово MERCI и тогда снова ищем M, предварительно прибавив к I величину, равную числу просмотренных букв.

Если же, напротив, найдено "MERCI" и цикл завершился нормально, то значение K не определено, и, следовательно, K надо присвоить 5. Поскольку нельзя встретить подряд два "MERCI", между которыми нет по крайней мере одного разделителя, значение K завышается на единицу.

Заметим, что для изменения значений I нельзя применить DØ-цикл, поскольку эта переменная в некоторых случаях увеличивается на K (инструкция с меткой 3), а в некоторых на 1, что запрещается делать внутри цикла, шаг которого определяется раз навсегда.

5.5. ПОСЛЕДНИЙ СПОСОБ ВЫЧИСЛЕНИЯ ЧИСЛА π

(*условие задачи на стр.* 58)

Ищем разделители, т. е. все литеры, не являющиеся буквами (от A до Z). К счастью, конструкторы вычислительных машин разумно выбрали представления для букв: они представляются связной совокупностью двоичных кодов, возрастающих от A к Z. Разделители всегда лежат вне этого интервала. Но могут встретиться подряд несколько разделителей

```
      INTEGER   P(320), CHIFF(9),A, Z
      DATA CHIFF /1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/,
     1  A,Z/1HA,1HZ/, BLANC /1H /
      READ 1,P
      PRINT 10, P
      N=1
      K=0
      DO 4 I=1,320
      IF (P(I) .LT. A .OR. P(I) .GT. Z) GO TO 2
      K=K+1
      GO TO 4
2     IF (K .EQ. 0) GO TO 4
      PRINT 3, (BLANC, J=1,N),CHIFF(K)
      K=0
      N=N+1
4     CONTINUE
      STOP
1     FORMAT (80A1)
3     FORMAT (1H+, 31A1)
10    FORMAT (4(1X,80A1/), //)
      END
```

```
QUE J'AIME A FAIRE APPRENDRE UN NOMBRE UTILE AUX SAGES,
IMMORTEL ARCHIMEDE, ARTISTE INGENIEUR,
QUI DE TON JUGEMENT PEUT PRISER LA VALEUR?
POUR MOI, TON PROBLEME EUT DE PAREILS AVANTAGES.

31415926535897932384626433832 79
```

Рис. 110

(например, два пробела). Таким образом, после того, как мы обнаружим один пробел, надо убедиться, что следующая литера — буква (эту роль как раз выполняет IF с меткой 2) (рис. 110).

Избранный способ печати, вообще говоря, не рекомендуется, потому что при нем каждая цифра печатается с помощью отдельной команды печати, и используется литера «+», которая запрещает переход к следующей строке.

Записав (BLANC, J = 1, N), получаем «печать» N пробелов, причем N увеличивается на единицу при каждой най-

денной цифре, во избежание того, чтобы она не наложилась на предыдущую при выводе.

Цифру приходится печатать как литеру, ибо нельзя использовать спецификацию «I», так как цифра последовательно перемещается вправо, что привело бы к постепенной модификации формата. Благодаря спецификации 31 A1 исключается модификация формата, но тогда список должен содержать одни лишь литеры. Необходимое преобразование кодов поризводится с помощью массива CHIFF.

Чтобы обойтись без литеры «+», запрещающей переход к следующей строке, нам пришлось переписать программу

```
      INTEGER   P(320), CHIFF(9),PI(31), A, Z
      DATA A,Z,PI,CHIFF/1HA,1HZ,31*1H  ,1H1,1H2,1H3,1H4,1H5,1H6,1H7,
     1  1H8,1H9/
      READ 1, P
      PRINT 10, P
      N=1
      K=0
      DO 4 I=1,320
      IF (P(I).LT.A.OR.P(I).GT.Z) GO TO 2
      K=K+1
      GO TO 4
2     IF (K.EQ.0) GO TO 4
      PI(N)=CHIFF(K)
      K=0
      N=N+1
4     CONTINUE
      PRINT 3, PI
      STOP
3     FORMAT (80A1)
2     FORMAT (//1X, 31A1)
10    FORMAT (1X, 80A1)
      END
```

Рис. 111

(рис. 111). На этот раз каждая найденная цифра пересылается во вспомогательный массив P. Затем можно осуществить более рациональную печать.

5.6. БУКВЕННАЯ ЗАПИСЬ СТА ПЕРВЫХ ЦЕЛЫХ ЧИСЕЛ

(*решение задачи на стр.* 59)

Все используемые в этом упражнении переменные определяются следующим образом:

U: цифра из разряда единиц,
D: цифра из разряда десятков,
N: «переводимое» число,
ZØNE: одномерный массив, в который засылается буквенная запись N,

UNITE: двумерный массив (2,20), элементы которого определяются в первой инструкции DATA следующим образом:

```
UNITE (1, 1), UNITE (2, 1)
UNITE (1, 2), UNITE (2, 2)      U N
UNITE (1, 3), UNITE (2, 3)      D E U X
.....
UNITE (1, 19), UNITE (2, 19)    D I X - H U I T
UNITE (1, 20), UNITE (2, 20)    D I X - N E U F
```

DIZN: двумерный массив (3,10), элементы которого [1]) определяются во второй инструкции DATA:

```
DIZN (1, 1), DIZN (2, 1), DIZN (3, 1)
DIZN (1, 2), DIZN (2, 2), DIZN (3, 2)
DIZN (1, 3), DIZN (2, 3), DIZN (3, 3)      D I X
DIZN (1, 4), DIZN (2, 4), DIZN (3, 4)      V I N G T
.....
DIZN (1, 7), DIZN (2, 7), DIZN (3, 7)      S Ø I X A N T E
DIZN (1, 8), DIZN (2, 8), DIZN (3, 8)      S Ø I X A N T E
DIZN (1, 9), DIZN (2, 9), DIZN (3, 9)      Q U A T R E - V I N G T
DIZN (1, 10), DIZN (2, 10), DIZN (3, 10)   Q U A T R E - V I N G T
```

Особенности массивов UNITE и DIZN станут понятны в дальнейшем. В настоящий момент заметим только, что четыре последних элемента второго массива попарно одинаковы. Остается лишь «добавить» к любому из них элемент из массива UNITE, чтобы получить правильное число: это почти что система с основанием 20. (В условии задачи эти числа объединены в 3-ю группу и строятся так: "SØIXANTE" и "DIX-HUIT" = 78.)

Числа, написанные буквами, заносятся в массив ZØNEX, который предварительно заполняется пробелами в цикле DØ2I. При пересылке в массив ZØNEX буквенная запись числа освобождается от лишних пробелов, которые, как мы увидим, могут в ней быть, когда она находится в массиве ZØNE. Поскольку массив ZØNEX должен вмещать 21 букву, нужно отвести для него 6 слов в машине IBM, 4 слова в машине UNIVAC и 3 — в CDC.

Начинаем с засылки названия десятка в ZØNE(1), ZØNE(2), ZØNE(3), причем D + 1 указывает на его место в массиве (рис. 112). Таким образом, если число заключено между 10 и 19, то засылаются пробелы, так как его истинное

[1]) Числа 10, 20, ..., 60, 60, 80, 80. — *Прим. ред.*

название надо будет взять из массива UNITE. Вслед за названием десятков надо поставить «ET», если число единиц равно 1 и само число лежит между 20 и 80 (тройное .AND. в программе). В этом случае в ZØNE(4) засылается цепочка литер "∧ET∧".

Если N = 80, то в ZØNE(4) надо заслать "S". Заметим, что ранее в ZØNE(4) на всякий случай были занесены пробелы, и поэтому, если не имеет место один из двух предыдущих случаев, форма написания остается правильной.

Следующий оператор IF разбивает числа на три группы, упомянутые в условии:

1-я группа: $1 < N < 19$
2-я группа: $20 < N < 60$
3-я группа: $61 < N < 99$

В двух крайних группах цифра единиц заключена между 1 и 19, во второй группе используется значение U, вычисленное ранее и заключенное между 1 и 10. Последний из трех последовательных операторов IF позволяет выбрать нужное значение U.

Но, поскольку в Фортране не допускаются переменные с нулевым индексом (который получился бы при N = 10, 20, 30, ..., 90), в правом индексе массива UNITE к U добавляется единица, и тогда, если U = 0, выбирается цепочка пробелов.

Вот некоторые случаи из тех, которые могут встретиться [1]):

```
^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  UN^ ^  ^ ^ ^ ^
^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  DIX^  ^ ^ ^ ^
VINGT^^  ^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^
VINGT^^  ^ ^ ^ ^  ^ ^ ^ ^  ET^ ^  UN^ ^  ^ ^ ^ ^
VINGT^^  ^ ^ ^ ^  ^ ^ ^ ^  ^ ^ ^ ^  QUAT RE^ ^
CINQUANTE^ ^ ^  ET^ ^  UN^ ^  ^ ^ ^ ^
CINQUANTE^ ^ ^  ^ ^ ^ ^  QUAT RE^ ^
SØIXANTE^ ^ ^  ET^ ^  ØNZE  ^ ^ ^ ^
QUAT RE-VINGT^ ^ ^ ^  UN^ ^  ^ ^ ^ ^
QUAT RE-VINGT^ ^ ^ ^  QUIN ZE^ ^
```

Затем следует убрать все ненужные пробелы. Фрагмент программы написан в соответствии с блок-схемой на рис. 113. Ищем первый пробел. Как только он найден, записываем его и переходим ко второму циклу, в котором ищется первая литера, отличная от пробела; при этом ничего не пересылается

[1]) Приводится запись чисел 1, 10, 20, 21, 24, 51, 54, 71 (60 и 11), 81, 95 ($4 \times 20 + 15$). — *Прим. ред.*

в ZØNEX, так как не встречается ничего, кроме пробелов, I — это индекс исходного массива (ZØNE), J — индекс массива, в который засылается текст, свободный от пробелов (ZØNEX).

```
C       ЧИСЛА С 1 ДО 99, НАПИСАННЫЕ БУКВАМИ
C
        INTEGER UNITE(2,20),DIZN(3,10),ZONE(6),ZONEX(6),U,D,S,BLANC,ET
        DATA UNITE/ 1H ,1H ,2HUN,1H ,4HDEUX,1H ,4HTROI,1HS,4HQUAT,2HRE,
     1  4HCINQ,1H ,3HSIX,1H ,4HSEPT,1H ,4HHUIT,1H ,4HNEUF,1H ,3HDIX,
     2  1H ,4HONZE,1H ,4HDOUZ,1HE,4HTREI,2HZE,4HQUAT,4HORZE,4HQUIN,2HZE
     3  ,4HSEIZ,1HE,4HDIX-,4HSEPT,4HDIX-,4HHUIT,4HDIX-,4HNEUF/
        DATA DIZN/1H ,1H ,1H ,1H ,1H ,1H ,4HVING,1HT,1H ,4HTREN,2HTE,
     1  1H ,4HQUAR,4HANTE,1H ,4HCINQ,4HUANT,1HE,4HSOIX,4HANTE,1H ,
     2  4HSOIX,4HANTE,1H ,4HQUAT,4HRE-V,4HINGT,4HQUAT,4HRE-V,4HINGT/
        DATA ET, S/3H ET,1HS/
C
        DO 1 N=1,99
        U=MOD(N,10)
        D=N/10
        DO 2 J=1,6
2       ZONEX(J)=DIZN(1,1)
        ZONE(1)=DIZN(1,D+1)
        ZONE(2)=DIZN(2,D+1)
        ZONE(3)=DIZN(3,D+1)
        ZONE(4)=DIZN(1,1)
        IF (N.EQ.80) ZONE(4)=S
        IF (N.GT.20.AND.N.LT.81.AND.U.EQ.1) ZONE(4)=ET
        IF (N.LT.20.OR.N.GT.60) U=MOD(N,20)
        ZONE(5)=UNITE(1,U+1)
        ZONE(6)=UNITE(2,U+1)
C
        I=1
        J=1
        CALL APLCAR(DIZN(1,1),1,BLANC)
        IF (N.LT.20) GO TO 7
C
9       CALL APLCAR(ZONE(1),I,MOT)
        IF (MOT.NE.BLANC) GO TO 6
        CALL SORCAR(ZONEX(1),J,MOT)
        J=J+1
7       I=I+1
        IF (I.EQ.25) GO TO 8
        CALL APLCAR(ZONE(1),I,MOT)
        IF (MOT.EQ.BLANC) GO TO 7
6       CALL SORCAR (ZONEX(1),J,MOT)
        J=J+1
        I=I+1
        IF (I.NE.25) GO TO 9
8       PRINT 5, N, ZONEX
1       CONTINUE
        STOP
5       FORMAT (I4,3X,6A4)
        END
```

Рис. 112

На блок-схеме есть «специальный вход», который необходим, чтобы исключить пробелы слева от названий чисел с 1 до 19 (они всегда начинаются двенадцатью пробелами). В этом случае и используется вход сразу во внутренний цикл,

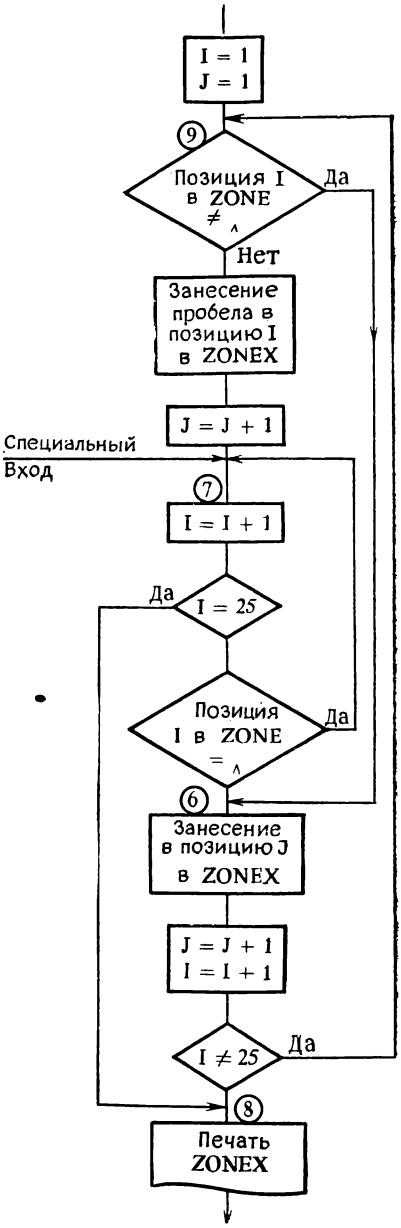

I = 1
J = 1
9
Позиция I в ZONE ≠ ^
Да
Нет
Занесение пробела в позицию I в ZONEX
J = J + 1
Специальный Вход
7
I = I + 1
Да
I = 25
Позиция I в ZONE = ^
Да
6
Занесение в позицию J в ZONEX
J = J + 1
I = I + 1
I ≠ 25
Да
8
Печать ZONEX

Рис. 113

чтобы устранить *всю совокупность* этих пробелов (IF(N.LT.20)GØTØ 7).

Для того чтобы литеры, выделенные подпрограммой APLCAR, сравнивались правильно, надо, чтобы пробелы имели одинаковый код. Поэтому «формируют» подходящий пробел с помощью предварительного обращения к подпрограмме APLCAR.

Последнее замечание: первая проверка I = 25, которая отсылает к метке 8, полезна, когда последняя литера в ZØNE — пробел. Если это не так, то выход из цикла обеспечивает вторая проверка (I ≠ 25).

5.7.1. СТАРАТЕЛЬНЫЙ НАБОРЩИК. ПОДГОТОВКА К РАБОТЕ

(*условие задачи на стр.* 59)

Все используемые переменные будут иметь тип INTEGER, благодаря инструкции IMPLICIT в начале программы (рис. 114).

Начинаем с исследования конца строки, чтобы узнать, какие действия надо предпринять. Найдем первую литеру, отличную от пробела, начиная справа и двигаясь влево.

Если это точка, то ничего делать не надо: RETURN.

Если это не точка и эта литера находится в (LØNG — 1)-й позиции, то надо только сдвинуть последнее слово справа на одну позицию вправо, что и делается инструкцией с меткой 10.

В противном случае мы должны уничтожить самое правое слово. Для этого надо последовательно определить:

- NCAR: число букв в этом слове + 1;
- DEBUT: номер позиции правой литеры первого слова;
- NBLANC: число вставляемых пробелов.

Затем продолжим просмотр справа налево и, когда найдем первый пробел, вычислим значение NCAR.

Чтобы найти значение DEBUT, будем просматривать текст слева направо и искать сначала первую литеру, не равную пробелу (чтобы пропустить возможные пробелы в начале строки), а затем номер позиции первого после нее пробела. Этот номер, уменьшенный на 1, даст значение DEBUT (в программе следует отнять 2 [1]), так как мы положили I = I + 1).

Затем возобновим просмотр справа налево с того места, где мы его прекратили. Этот просмотр прекратится, когда K

[1]) Это существенно, по-видимому, только в одном случае, когда в строке всего два слова и одно из них исключается. Но эта особая ситуация в данном упражнении вообще не рассматривается. — *Прим. ред.*

```
      SUBROUTINE JUSTIF (LIGNE, NCAR, LONG)
      IMPLICIT INTEGER (A-Z)
      DIMENSION LIGNE (1)
      DATA BLANC /1H /, POINT /4H   ./
      NBLANC=1
      NCAR=0
      K=LONG
4     CALL APLCAR (LIGNE, K, M)
      K=K-1
      IF (M.EQ.BLANC) GO TO 4
      IF (M.EQ.POINT.AND.K.NE.LONG-2) RETURN
      IF (K.EQ.LONG-2) GO TO 10
      K=LONG-1
1     CALL APLCAR (LIGNE, K, M)
      K=K-1
      IF (M.NE.BLANC) GO TO 1
      NCAR=LONG-K
      I=1
6     CALL APLCAR (LIGNE, I, M)
      I=I+1
      IF (M.EQ.BLANC) GO TO 6
7     CALL APLCAR (LIGNE, I, M)
      I=I+1
      IF (M.NE.BLANC) GO TO 7
      DEBUT=I-1
      J=K-1
      J=K
3     K=K-1
      IF (K.EQ.DEBUT) GO TO 2
      CALL APLCAR (LIGNE, K, M)
      IF (M.NE.BLANC) GO TO 3
      NBLANC=NBLANC+1
      GO TO 3
2     Q=NCAR/NBLANC
      R=MOD(NCAR, NBLANC)
      I=LONG
11    CALL APLCAR (LIGNE, J, M)
      CALL SORCAR (LIGNE, I, M)
      I=I-1
      IF (M.NE.BLANC) GO TO 8
      IF (Q.EQ.0) GO TO 5
      DO 9 K=1,Q
      CALL SORCAR (LIGNE, I, BLANC)
9     I=I-1
5     IF (R.EQ.0) GO TO 8
      CALL SORCAR (LIGNE, I, BLANC)
      R=R-1
      I=I-1
8     J=J-1
      IF (J.GE.DEBUT) GO TO 11
      RETURN
C
10    NCAR=1
      K=LONG-1
12    CALL APLCAR(LIGNE, K, M)
      CALL SORCAR (LIGNE, K+1, M)
      IF (M.EQ.BLANC) RETURN
      K=K-1
      GO TO 12
      END
```

Рис. 114

будет равно DEBUT. Каждый раз, когда встречается пробел, значение NBLANC увеличивается на единицу.

Теперь надо распределить NCAR пробелов в NBLANC мест. Делим NCAR на NBLANC, что дает Q — число пробелов, добавляемых к каждому уже имеющемуся пробелу. Но вообще NCAR не будет в точности кратно NBLANC. Остаток от деления, R, полученный с помощью операции MØD, указывает, к скольким промежуткам между словами, начиная справа, надо добавить по лишнему пробелу.

Все эти предварительные вычисления позволяют приступить к перемещениям. Сначала по одной литере переносится предпоследнее слово в самый правый край строки. Конец этого слова обозначен пробелом. J — номер литеры в прежней конфигурации, I — ее новый порядковый номер.

Когда встречается пробел, мы пересылаем его, и теперь, не изменяя J, надо вставить Q дополнительных пробелов. Это делается в цикле DØ 9 K. Если значение R не равно нулю, пересылают еще один пробел. При каждой пересылке пробела I уменьшается на единицу. Поскольку R указывает, в скольких местах мы должны добавить дополнительные пробелы, его также надо уменьшить. Этот цикл повторяется до J = = DEBUT.

Известно, что DØ-циклы выполняются по крайней мере один раз. Если Q=0, нужна проверка, чтобы избежать входа в цикл. Остается обработать случай, когда имеется всего один пробел в правом краю. Тогда сдвигается лишь последнее слово на одну позицию вправо, и это делается в цикле, первая инструкция которого имеет метку 10. Цикл прекращается при обнаружении первого пробела.

Другим неудобством DØ-циклов (не во всех машинах, но по крайней мере в IBM 360/370) является невозможность использования отрицательных шагов. Вот почему некоторые повторения здесь осуществляются без помощи DØ-цикла.

Замечание. Величине NCAR присваивается значение нуль в начале программы, на тот случай, если в результате выполнения инструкции

```
IF (M.EQ.PØINT) RETURN
```

тотчас же происходит возврат в вызывающую программу.

5.7.2. СТАРАТЕЛЬНЫЙ НАБОРЩИК. ВЫПОЛНЕНИЕ РАБОТЫ

(*условие задачи на стр.* 60)

Текст, который надо обработать, вводится с 13 перфокарт. Затем для печати он разбивается на «куски» по 100 литер.

Печать исходного текста полезна при проверке правильности программы. В программе на рис. 115 обратите внимание на инструкцию FØRMAT с меткой 3, которая перед на-

```
      DIMENSION TEXTE (250), Z(33)
      LOGICAL B
      DATA BLANC /1H /
      READ 1, TEXTE
      PRINT 3, TEXTE
1     FORMAT (20A4)
3     FORMAT(/////10(1X,25A4/))
8     READ (5, 20, END=2) LIGNE
20    FORMAT (I3)
      PRINT 21, LIGNE
21    FORMAT (/////10X, 8HСТРОКА ИЗ, I4, 12H ЛИТЕР.
      NLIGNE=LIGNE/4
      I=0
7     J=1
      B=.TRUE.
      DO 10 K=1,33
10    Z(K)=BLANC
12    I=I+1
      IF (I.GT.1000) GO TO 8
      CALL APLCAR (TEXTE, I, A)
      IF (A.NE.BLANC) B=.FALSE.
      CALL SORCAR (Z, J, A)
      J=J+1
      IF (J.LE.LIGNE) GO TO 12
      IF (B) GO TO 6
      CALL APLCAR (TEXTE, I+1, A)
      IF (A.EQ.BLANC) GO TO 6
      CALL JUSTIF (Z, NC, LIGNE)
      I=I-NC+1
6     PRINT 4, (Z(K),K=1,NLIGNE)
      IF (A.EQ.BLANC.AND..NOT.B) I=I+1
      GO TO 7
2     STOP
4     FORMAT (1X33A4)
      END
```

Рис. 115

печатанным текстом пропустит 5 пустых строк. Если бы мы написали

(/////(1X25A4/)/),

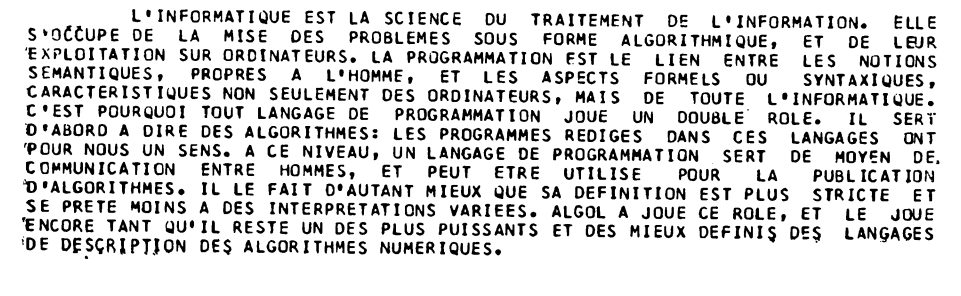
то получили бы

5 пустых строк,
1 строку текста,
2 пустых строки,
1 строку текста,
2 пустых строки
и т. д.

В самом деле, после того как напечатается одна строка текста, «выполнятся» две косые черты, и, поскольку список

еще не исчерпан, просмотр формата продолжится со второй открывающей скобки.

Если же мы написали так, как это сделано в нашей программе, повторный просмотр формата не потребуется, потому



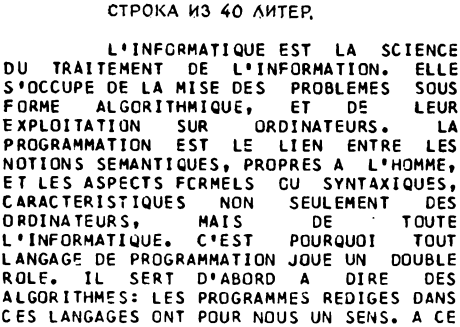


Рис. 116

что его длина равна длине обрабатываемого списка ($10 \times 25 = 250$). Таким образом, надо не забыть только о косой черте в повторяющейся группе, чтобы осуществлялся переход к новой строке. Последняя косая черта служит всего один раз, когда весь текст уже напечатан.

Индекс I ($1 \leqslant I \leqslant 1000$) указывает на литеру, которая пересылается из массива TEXTE. Второй индекс J ($1 \leqslant J \leqslant$

$\leqslant$ LIGNE) используется для заполнения массива Z с помощью подпрограмм APLCAR — SØRCAR. Подпрограмме JUSTIF нельзя в качестве строки передавать массив TEXTE, так как во время выравнивания теряются куски слов. В ходе пересылок подсчитывается число пробелов, так как подпрограмма JUSTIF не умеет обрабатывать строки, целиком состоящие из пробелов. Если попадается такая строка, то она просто печатается (можно было бы ограничиться интервалом между строками с форматом типа (1X), что, безусловно, привело бы к выигрышу времени, так как отсутствовал бы список в инструкции PRINT).

После заполнения массива Z индекс I устанавливается на литеру в массиве TEXTE, которая должна была бы обрабатываться следующей.

Прежде чем вызвать JUSTIF, необходимо проверить, не является ли пробелом следующая литера TEXTE. Если да, то не следует вызывать подпрограмму JUSTIF, которая понапрасну уничтожила бы последнее слово (в этом случае оно полное). После возврата из JUSTIF индекс I уменьшается на величину, равную числу уничтоженных литер, и массив Z печатается.

Если первая литера новой строки — пробел, то I надо увеличить на единицу, чтобы эта строка не начиналась с пробела, конечно, за исключением случая, когда она вся состоит из пробелов.

Работа продолжается, пока I меньше 1000. Действительно, последняя строка не может содержать меньше двух литер, поскольку она кончается точкой, а точка следует непосредственно за последним словом. В самом деле, в условии задачи подчеркивалось, что текст составлен из фраз при строгом соблюдении обычных правил пунктуации и набора.

На рис. 116 даются два примера применения программы: для строки с 40 и с 80 литерами. Можно сравнить результат, полученный в последнем случае, с картами данных на рис. 31 (стр. 61).

6. ОБРАБОТКА ФАЙЛОВ

6.1. СЛИЯНИЕ ДВУХ ФАЙЛОВ

(*условие задачи на стр.* 63)

Решение организовано в соответствии с блок-схемой на рис. 117, соответствующая программа воспроизведена на рис. 118.

Программа начинается с чтения одной записи из F1 в Z1 и одной записи из F2 в Z2; затем присваивается 1 счетчикам

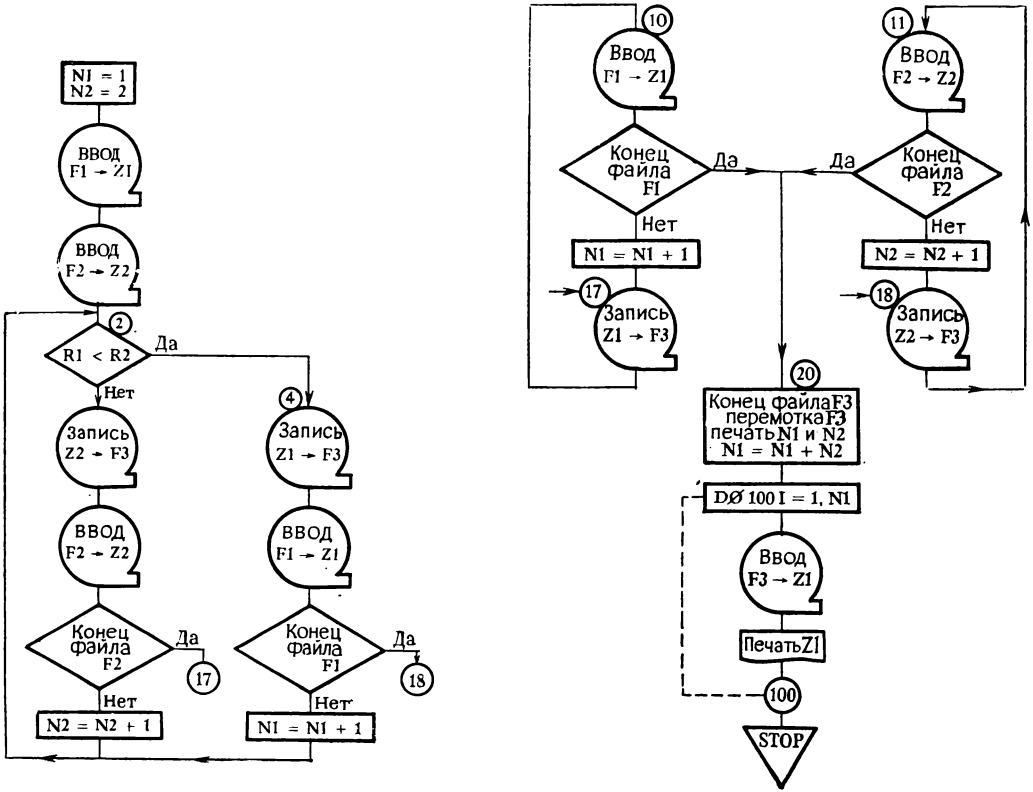

N1 = 1
N2 = 2
ВВОД F1 → Z1
ВВОД F2 → Z2
2
R1 < R2
Да
Нет
Запись Z2 → F3
ВВОД F2 → Z2
Конец файла F2
Да
17
Нет
N2 = N2 + 1
4
Запись Z1 → F3
ВВОД F1 → Z1
Конец файла F1
Да
18
Нет
N1 = N1 + 1
10
ВВОД F1 → Z1
Конец файла F1
Да
Нет
N1 = N1 + 1
17
Запись Z1 → F3
11
ВВОД F2 → Z2
Конец файла F2
Да
Нет
N2 = N2 + 1
18
Запись Z2 → F3
20
Конец файла F3
перемотка F3
печать N1 и N2
N1 = N1 + N2
DØ 100 I = 1, N1
ВВОД F3 → Z1
Печать Z1
100
STOP

Рис. 117

N1 и N2 (только что считали по одной записи из каждого файла).

Поскольку по условию задачи записи сформированы из литер, нужен такой формат ввода, который отражает общую структуру файлов. Единственное поле в записи, которое нас интересует при слиянии, это ключ (Z1(1) и Z2(1)), все остальное просто-напросто переписывается.

```
      INTEGER Z1(9), Z2(9)
      N1=1
      N2=1
      READ (8,1) Z1
      READ (10,1) Z2
2     IF (Z1(1) .LE. Z2(1)) GO TO  4
      WRITE (11,1) Z2
      READ(10,1,END=17) Z2
      N2=N2+1
      GO TO 2
4     WRITE (11,1) Z1
      READ (8,1,END=18) Z1
      N1=N1+1
      GO TO 2
C     КОНЕЦ ФАЙЛА
10    READ (8,1,END=20) Z1
      N1=N1+1
17    WRITE (11,1) Z1
      GO TO 10
C
11    READ (10,1,END=20) Z2
      N2=N2+1
18    WRITE (11,1) Z2
      GO TO 11
C     КОНЕЦ ПОСЛЕДНЕГО ФАЙЛА
20    END FILE 11
      REWIND 11
      PRINT 40, N1, N2
      N1=N1+N2
      DO 100 I=1,N1
      READ(11,1) Z1
100   PRINT 41, Z1
      STOP 123
1     FORMAT (I8,8A4)
40    FORMAT(I4,30H ЗАПИСЕЙ ПРОЧИТАНО ИЗ F1 И ,I4,27H ЗАПИСЕЙ ПРОЧ
     1 ,ИТАНО ИЗ F2,///)
41    FORMAT (1X I8, 8A4)
      END
```

Рис. 118

Таким образом, в программе будут две аналогичные ветви, выбираемые в соответствии с результатом сравнения ключей и отличающиеся друг от друга лишь номером входного файла. Записи в обоих случаях помещаются в один и тот же выходной файл.

В случае если ключи оказались одинаковыми, можно считать, что запись из F1 имеет более высокий приоритет, чем запись из F2, и они переписываются в F3 с учетом приоритета. Снова читается файл, из которого копировалась последняя

запись. При каждом считывании увеличивается счетчик записей, соответствующий входному файлу (N1 или N2).

Этот процесс продолжается до тех пор, пока не обнаружится конец файла F1 или F2. Затем управление передается одному из двух участков в программе, отличающихся друг от друга лишь номером входного файла и предназначенных для копирования оставшегося файла в F3. Не следует забывать о копировании последней записи, прочитанной из этого файла (F1 или F2) и все еще находящейся в памяти.

Встреченный затем при чтении конец файла указывает на то, что слияние закончено. Выходной файл закрывается, делается обратная перемотка, и он снова читается в DØ-цикле и печатается. При печати используется формат, отличающийся от формата чтения или записи лишь пробелом, который управляет переходом к следующей строке.

ВОПРОС:

Почему не делается проверка (END=) конца файла, когда заново читается F3 для печати?

ОТВЕТ:

Потому что известно число записей (N1), введенных из F1 и (N2) из F2, и хорошо известно, что вновь будет считано не более N1 + N2 записей.

6.2. СЛИЯНИЕ n ФАЙЛОВ

(*условие задачи на стр.* 63)

Как отмечалось в условии, мы намеренно ограничились не более, чем 4 файлами. Для входных файлов выбираем номера логических устройств 8, 9, 10 и 11, а для выходного файла номер 1. Кроме того, определяем массив T(4,20), в который попадут:

в строку	1	записи	с логического	устройства	8
—	2	—	—	—	9
—	3	—	—	—	10
—	4	—	—	—	11

Константа с именем BIG содержит число, превосходящее самый большой возможный ключ (в нашем случае 999999999) (см. рис. 119). Наконец, вводим карту, содержащую NFIC — число файлов, которые надо слить. Проверяется правильность NFIC, и если оно меньше 1 или больше 4, программа останавливается на инструкции STØP 10.

В том случае, когда NFIC окажется равным 1 (тогда речь идет о простом копировании), чтобы не нарушать общность, T(2,1) присваивается значение константы BIG.

```
      INTEGER   T(4,20),TT(20),BIG
      EQUIVALENCE(TT(1),T(1,1))
      DATA BIG/999999999/
C     СОЗДАТЬ 4 ФАЙЛА.
      DATA Z/4H ABC/
      DO 69 I=3,20
69    TT(I)=Z
      TT(2)=8
      DO 70 I=1,10
      TT(1) =I*5
70    WRITE(8) TT
      TT(2)=9
      DO 71 I=1,24
      TT(1)=I*2
71    WRITE(9) TT
      TT(2)=10
      DO 72 I=1,8
      TT(1)=I*3
72    WRITE(10) TT
      TT(2)=11
      DO 73 I=1,14
      TT(1)=I*4
73    WRITE(11) TT
      ENDFILE 8
      ENDFILE 9
      ENDFILE 10
      ENDFILE 11
      REWIND 8
      REWIND 9
      REWIND 10
      REWIND 11
C           НАЧАЛО ПРОГРАММЫ СЛИЯНИЯ.
      READ 120,NFIC
      IF(NFIC.LT.1.OR.NFIC.GT.4) STOP 10
      T(2,1)=BIG
      DO 1 I=1,NFIC
      IA7=I+7
1     READ(IA7) (T(I,J),J=1,20)
      NF=NFIC
9     N1=1
      DO 2 I=2,NFIC
      IF(T(N1,1).GT.T(I ,1)) N1=I
2     CONTINUE
      WRITE(1) (T(N1,J),J=1,20)
      IA7=N1+7
      READ(IA7,END=50) (T(N1,J),J=1,20)
      GO TO 9
50    NF=NF-1
      T(N1,1)=BIG
      IF(NF.NE.0) GO TO 9
      ENDFILE 1
      REWIND 1
53    READ(1 ,END=52) TT
      PRINT 100,TT
      GO TO 53
52    STOP 2
100   FORMAT(2I5,18A4)
120   FORMAT(I2)
      END
```

Рис. 119

Затем в первые NFIC строк массива T считываются первые записи из соответствующих NFIC сливаемых файлов. NF принимает значение NFIC (эта переменная в дальнейшем будет служить для подсчета числа встреченных концов файлов и передачи управления на конец программы, когда встретится NFIC концов файлов).

Теперь все готово к началу слияния. N1 принимает значение 1, I — значение 2, и переходим к сравнению ключа N1-й строки массива T с ключом I-й строки T. Если последний меньше, то переменной N1 присваивается значение I и то же самое повторяется снова после прибавления 1 к I; процесс продолжается до тех пор, пока I не достигнет значения NFIC. Эти действия выполняются в цикле DØ2I. При выходе из цикла N1 указывает на наименьший из NFIC ключей. Если N1 = 1, тогда в T(I, 1), т. е. в ключе записи, введенной с логического устройства 9 (которое в рассматриваемом случае не существует), находится самый большой возможный код и N1 сохраняет значение 1, что совершенно корректно.

Затем на логическое устройство 1 записывается N1-я строка массива T, которая затем «перезаряжается» путем считывания следующей записи с логического устройства (8 + N1 — — 1), т. е. N1 + 7.

Если в этот момент встречается конец файла, из NF вычитается 1 и первому элементу N1-й строки T присваивается значение BIG, потому что осталось всего NF непустых файлов. Когда NF принимает значение 0, слияние заканчивается.

Если же конец файла не встретился в момент считывания или NF отлично от нуля, продолжается работа с N1 = 1. Повторное считывание и печать выходного файла не представляют никакой трудности.

ВОПРОСЫ:

1. Какова роль оператора EQUIVALENCE в начале программы?
2. Почему не написали IF(NFIC.EQ.1)T(2, 1) = BIG, если это существенно лишь при NFIC = 1?
3. Почему нет номера оператора FØRMAT в инструкциях ввода-вывода?

ОТВЕТЫ:

1. Экономия памяти ... и времени! В самом деле, чтобы сэкономить память, достаточно было бы читать выходной файл в T(1, *) или T(2, *). Но тогда понадобится неявный DØ-цикл. Возможность просто написать TT даст вообще ощутимый выигрыш при чтении и записи.

2. Если NFIC больше 1, константа BIG будет заменена подлинным ключом. Таким образом экономим одну систематическую проверку.
3. Поскольку записи сделаны в «двоичной системе», это избавляет нас от потерь на преобразования.

6.3. КОРРЕКТИРОВКА ФАЙЛА НА МАГНИТНОЙ ЛЕНТЕ

(*условие задачи на стр.* 64)

Блок-схему решения этой задачи можно найти на рис. 120, а программу — на рис. 121.

Начинаем с чтения основной ленты B1, затем корректировочной ленты B2. Если номер товара с B1 меньше номера товара с B2, на ленту результатов B3 копируем запись с B1. Затем вновь читается B1, но при этом, разумеется, не читается следующая запись с B2, и опять сравниваются номера товара. Если товары с B1 и с B2 имеют одинаковые номера, проверяется код B2.

Если код = 3, то это означает, что надо изъять товар, прочитанный с ленты B1, и нас просто более не интересуют записи, только что прочитанные с B1 и с B2.

Если код = 1, то на ленту B3 копируется запись, прочитанная с ленты B2, потому что она должна заменить запись с B1. Затем следует переход в начало программы.

Если код = 2, изменяется запас товара, прочитанного с ленты B1, помещается на ленту B3 и снова следует переход в начало программы.

Если номер товара с B1 меньше номера товара с B2, это означает, что в соответствующей записи с B2 в коде должна стоять 1. Если это так, должна быть составлена запись, соответствующая новому товару. В этом случае копируем запись с B2 на B3 и возвращаемся к чтению B2. Если в коде не 1, то, как это было сказано в условии, программа останавливается.

Заметим, что в начале программы небезразлично, что читается раньше, B1 или B2. На блок-схеме (рис. 120) видно, что читается либо B1 и B2, либо только B2.

В случае когда обнаруживается конец файла, программа идет по двум возможным путям:

• Инструкция с меткой 58 производит обработку конца файла B1. Здесь все время читается B2 и переписывается на B3, причем выбираются лишь те товары, у которых соответствующие им записи имеют в коде 1, поскольку основной файл уже прочитан и к нему можно только добавить записи, соответствующие новым товарам. Любые изменения или изъятия не имеют больше никакого смысла. Когда обнаруживает-

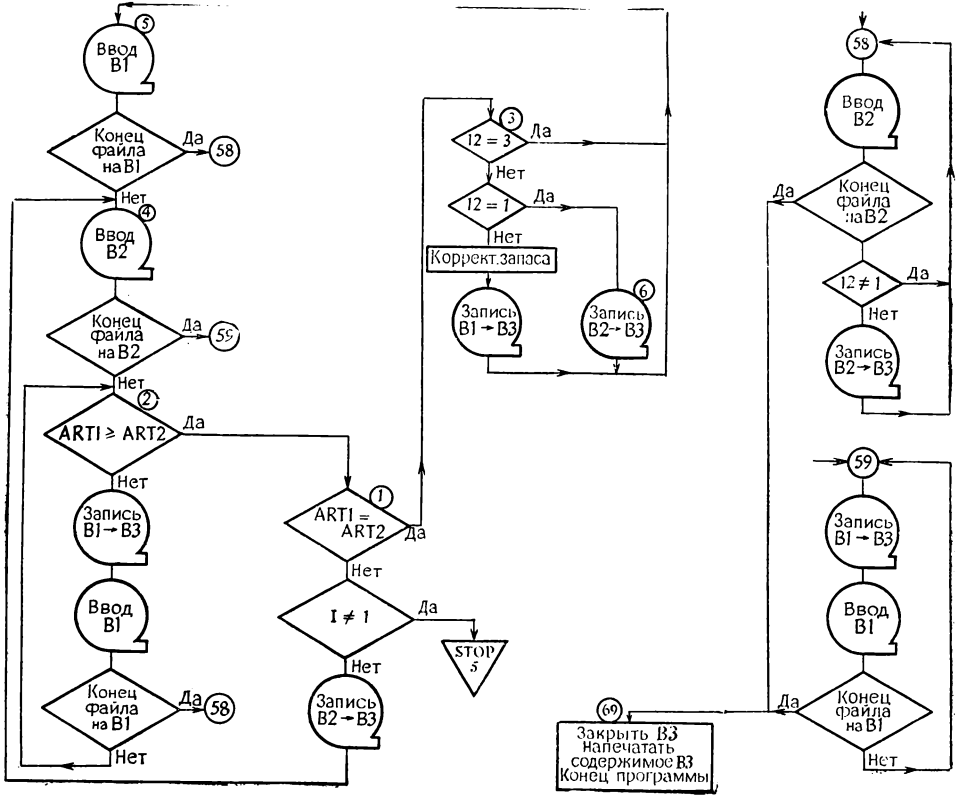

5
Ввод B1
Конец файла на B1
Да
58
Нет
4
Ввод B2
Конец файла на B2
Да
59
Нет
2
ART1 ≥ ART2
Да
Нет
Запись B1→B3
Ввод B1
Конец файла на B1
Да
58
Нет
1
ART1 = ART2
Да
Нет
I ≠ 1
Да
STOP 5
Нет
Запись B2→B3
3
I2 = 3
Да
Нет
I2 = 1
Да
Нет
Коррект.запаса
Запись B1→B3
6
Запись B2→B3
58
Ввод B2
Конец файла на B2
Да
I2 ≠ 1
Да
Нет
Запись B2→B3
59
Запись B1→B3
Ввод B1
Конец файла на B1
Да
Нет
69
Закрыть B3 напечатать содержимое B3 Конец программы

Рис. 120

ся конец файла на B2, выполняется заключительная часть программы, которая снова читает файл с ленты B3.

• Инструкция с меткой 59 производит обработку конца файла на ленте B2. Прежде всего копируется на B3 последняя запись, прочитанная с B1, затем переписывается без вся-

```
      DIMENSION NUM1(8),NUM2(8)
    5 READ(8,100,END=58) I1,NUM1
  100 FORMAT(I1,I9,5A4,I6,F6.2)
    4 READ(9,100,END=59)I2,NUM2
    2 IF(NUM1(1).GE.NUM2(1))GO TO 1
      WRITE(10,100)I1,NUM1
      READ(8,100,END=58) I1,NUM1
      GO TO 2
    1 IF(NUM1(1).EQ.NUM2(1)) GO TO 3
      IF(I2.NE.1) STOP 5
      WRITE(10,100)I1,NUM2
      GO TO 4
    3 IF(I2.EQ.3) GO TO 5
      IF(I2.EQ.1) GO TO 6
      NUM1(7)=NUM1(7)-NUM2(7)
      WRITE(10,100) I1,NUM1
      GO TO 5
    6 WRITE(10,100)I1,NUM2
      GO TO 5
   58 READ(9,100,END=69)I2,NUM2
      IF(I2.NE.1) GO TO 58
      WRITE(10,100)I2,NUM2
      GO TO 58
   59 WRITE(10,100)I1,NUM1
      READ(8,100,END=69)I1,NUM1
      GO TO 59
   69 ENDFILE 10
      REWIND 10
  501 READ(10,100,END=500)I1,NUM1
      PRINT 101,I1,NUM1
  101 FORMAT(10X,I1 I9,5A4,I6,F6.2)
      GO TO 501
  500 STOP
      END
```

Рис. 121

кого контроля файл с B1 на B3. В самом деле, больше невозможны никакие модификации. Наконец, выполняется та же программа печати B3, что и выше.

ВОПРОСЫ:

1. Создается впечатление, что в этой программе есть ошибка. Действительно, несколько раз к переменной типа INTEGER (NUM1 или NUM2) применяется спецификация, предназначенная для обработки вещественных чисел: F6.2.

2. Почему не используется массив NUM1 из 9 элементов? Ведь тогда можно было бы писать NUM1(1) вместо I1?

ОТВЕТЫ:

1. Вовсе нет! Это было бы ошибкой, если бы производились *вычисления* с NUM1(8) и NUM2(8), но мы их только читаем и записываем. Известно, что при выполнении программы не делается никаких проверок типа переменных в списке на соответствие спецификациям в операторе FØRMAT.
2. Проще использовать для кода скалярную переменную, чем массив. Чем меньше пишется литер, тем меньше риск их забыть (осторожно со скобками!). Можно также получить экономию времени, так как не надо вычислять индексы.

7. ОБО ВСЕМ ПОНЕМНОГУ...

7.1. 1 2 3 4 5 6 7 8 9 = 100

(*условие задачи на стр.* 65)

Для каждой новой цифры I имеется выбор между четырьмя инструкциями

```
1 | Z = Z + I
2 | Z = Z - I
3 | Z = Z * I
4 | Z = Z/I
```

Начальное значение Z равно 1 — это первая цифра слева. Естественно приходит мысль об использовании вычисляемого GØ TØ

```
GØTØ (1, 2, 3, 4), M
```

где M принимает значение 1 для сложения, 2 для вычитания и т. д.

В наши цели не входит оптимизация решения, поэтому мы пишем

```
    | Z = 1
    | DØ 10 K1 = 1, 4
    | GØ TØ (1, 2, 3, 4), K1
 1  | Z = Z + 2
    | GØ TØ 5
 2  | Z = Z - 2
    | GØ TØ 5
 3  | Z = Z * 2
    | GØ TØ 5
 4  | Z = Z/2
 5  | DØ 10 K2 = 1, 4
    | GØ TØ (11, 21, 31, 41), K2
11  | Z = Z + 3
    | GØ TØ 15
    | ... etc...
```

Мы не будем продолжать, обескураженные однообразием этого решения. Если ничто не вынуждает нас оптимизировать программу в отношении времени счета, то само собой разумеется, что следует поискать способ, позволяющий уменьшить число инструкций.

Можно было бы организовать вычисления иначе:

```
      |  Z = 1
      |  DØ 10 K1 = 1, 4
      |  DØ 10 K2 = 1, 4
      |    ...
      |  DØ 10 K8 = 1, 4
      |  GØ TØ (1, 2, 3, 4), K1
      |    ...
    5 |  GØ TØ (11, 21, 31, 41), K2
      |    ...
   15 |
      |    ...
      |  IF (Z .NE. 100) GØ TØ 10
      |  PRINT ...
   10 |  CØNTINUE
```

Число инструкций не изменилось, но их расположение показывает возможность сгруппировать все вычисляемые GØ TØ в один цикл. Получаем

```
      |  Z = 1
      |  DØ 10 K(1) = 1, 4
      |  DØ 10 K(2) = 1, 4
      |    ...
      |  DØ 10 K(8) = 1, 4
      |  DØ 11 I = 1, 8
      |  GØ TØ (1, 2, 3, 4), K(I)
    1 |  Z = Z + (I + 1)
    2 |    ...
    3 |    ...
    4 |    ...
   11 |  CØNTINUE
      |  IF (Z .NE. 100) GØ TØ 10
      |  PRINT ...
   10 |  CØNTINUE
```

Ни один компилятор не допускает использования переменных с индексами в DØ-цикле или в вычисляемом операторе

GØ TØ. Эту трудность нам поможет разрешить снова инструкция EQUIVALENCE:

```
      CØMMØN K1, K2, K3, K4, K5, K6, K7, K8
      DIMENSIØN K(8)
      EQUIVALENCE (K1, K(1))
```

Инструкция CØMMØN вынуждает компилятор отвести для K1, K2, ..., K8 совокупность последовательных слов. Инструкция EQUIVALENCE требует, чтобы K1 и K(1) зани-

```
C         1+-*/2+-*/3+-*/4+-*/5+-*/6+-*/7+-*/8+-*/9 = 100
          DIMENSION K(8), A(4), B(9)
          COMMON K1,K2,K3,K4,K5,K6,K7,K8
          EQUIVALENCE (K(1), K1)
          DATA A /1H+,1H-,1H*,1H//, EGAL /1H=/
          B(9)=EGAL
C
          DO 10 K1=1,4
          DO 10 K2=1,4
          DO 10 K3=1,4
          DO 10 K4=1,4
          DO 10 K5=1,4
          DO 10 K6=1,4
          DO 10 K7=1,4
          DO 10 K8=1,4
C
          Z=1
          DO 11 I=2,9
          M=K(I-1)
          GO TO (1,2,3,4),M
1         Z=Z+I
          GO TO 11
2         Z=Z-I
          GO TO 11
3         Z=Z*I
          GO TO 11
4         Z=Z/I
11        CONTINUE
          IF (Z.NE.100.) GO TO 10
C
          DO 12 I=1,8
          M=K(I)
12        B(I)=A(M)
          PRINT 5, (I,B(I),I=1,9)
5         FORMAT (1X 9(I2,1X,A1),4H 100 //)
10        CONTINUE
          STOP
          END
```

Рис. 122

мали одну и ту же ячейку; то же самое происходит соответственно с K2 и K(2), K3 и K(3) и т. д.

Таким образом, получаем решение, приведенное на рис. 122. Параметр цикла DØ11 изменяется от 2 до 9, а не

от 1 до 8, чтобы проще обозначались цифры, которые последовательно участвуют в вычислении. Значение K(I — 1) указывает, какой оператор ставится между I и Z.

При печати используются четыре «литерные» переменные A, и нам удобно было бы написать

```
|  | PRINT 5, (I, A(K(1)), I = 1,9)
```

но имеются два затруднения:

1. двойная индексация допускается далеко не всеми компиляторами,

2. K(9) не определено (имеются 9 цифр, но только 8 операторов).

```
1 + 2 + 3 + 4 + 5 * 6 - 7 + 8 + 9 = 100

1 + 2 * 3 + 4 + 5 * 6 - 7 + 8 - 9 = 100

1 * 2 + 3 * 4 - 5 * 6 - 7 + 8 + 9 = 100

1 * 2 + 3 * 4 * 5 + 6 - 7 - 8 + 9 = 100

1 * 2 + 3 * 4 * 5 - 6 + 7 + 8 - 9 = 100

1 * 2 - 3 + 4 * 5 * 6 - 7 + 8 + 9 = 100

1 * 2 * 3 + 4 + 5 * 6 - 7 + 8 + 9 = 100

1 / 2 + 3 * 4 * 5 + 6 + 7 + 8 + 9 = 100
```

Рис. 123

Первое затруднение разрешается при помощи вспомогательного массива B из 3 элементов, при этом начальным значением B(9) является «=», что одним ударом разрешает и второе затруднение.

ВОПРОСЫ:

1. Почему бы прямо так и не писать:

```
DATA B(9) /1H=/ ?
```

2. Разве необходимо использовать вещественную переменную Z, чтобы производить вычисления над целыми числами?

ОТВЕТЫ:

1. Некоторые компиляторы не допускают частичного присвоения начальных значений массиву с помощью инструкции DATA.

2. При делении мы рискуем получить нецелый результат. Тогда сравнение будет неверным (см. последнее решение, рис. 123).

7.2. ПЛАВАЮЩАЯ ФИКСИРОВАННАЯ ЗАПЯТАЯ

(*условие задачи на стр.* 65)

Результат должен быть представлен в виде цепочки литер. Мы используем массив IT для передачи в качестве аргумента представления числа в виде литер, среди которых могут быть 10 цифр из массива ICH, определенного инструкцией DATA, и знаки пунктуации —, ^, . и * (рис. 124).

Вначале заполняем массив IT пробелами в цикле DØ4. (В дальнейшем станет понятно, почему во вторую позицию на всякий случай ставится «.».)

Чтобы не нарушать правила, которое не рекомендует изменять значения параметров, передаваемых вызывающей программой, LL пересылается в L. Далее, если оказалось, что значение не заключено между 3 и 8 (см. условие задачи), оно принудительно делается равным 8. Если A равно нулю, в цикле DØ3 формируется результат, состоящий из:

.000 ... 0
(L — 2 нулей)

Точка уже была поставлена раньше.

В общем случае, когда A отлично от нуля, вычисляется L2 = L — 2, число цифр, которые нужно сформировать (резервируется место для знака и место для точки).

Десятичный логарифм *абсолютного значения* A (в случае, если A окажется отрицательным), увеличенный на 1, дает число десятичных знаков в целой части A. Если J превосходит L2 (максимальное число цифр, которое можно напечатать), надо составить последовательность из L звездочек, что делается после перехода на метку 20 в цикле DØ21.

Если A отрицательно, первой литерой в IT должен быть знак «—» (IT(1) = IM). Если J больше 0, то предстоит обрабатывать случай, когда A наверняка имеет целую часть и, возможно, дробную часть. Если нет, то некоторые начальные установки (которые мы пока оставляем в стороне) позволят снова вернуться к общему случаю.

После перехода на метку 10 определяется N = 2 и тем самым задается номер первой формируемой литеры в IT, затем определяются JJ и JK, соответственно начальное и конечное значения параметра цикла DØ 11, в котором последовательные цифры числа A заносятся в массив IT.

Инструкция с меткой 1 дает целое число K, состоящее из всех цифр A, которые надо напечатать вне зависимости от их положения относительно запятой. Например, если A=123,456 при L = 7, надо напечатать 5 цифр, и мы получаем K=12346 («+.5» используется для округления) [1]).

Теперь надо извлекать одну за другой, начиная слева, каждую из дробных цифр K и засылать эквивалентные им литеры (взятые из ICH) в последовательные элементы IT, определяемые значением N. Это делается в цикле DØ 11. Всегда, когда имеется целая часть A, этот цикл надо выполнить J раз (что следует из значений, присвоенных выше JJ и JK), пока не дойдем до десятичной точки.

Выражение 10 ** (L2 — M) дает последовательные степени 10, на которые делится K. В единичном разряде частного побывает последовательно каждая из цифр K, начиная слева. Благодаря коррекции K в частном всегда получается только одна цифра единиц, которая после прибавления 1 (чтобы учесть особенности адресации массива ICH) позволяет поместить соответствующую литеру в IT(N). N увеличивается на единицу при каждом повторении цикла. При выходе из цикла можно в случае необходимости заслать десятичную точку.

Значение JJ равно 1, если обрабатывалась целая часть A; в противном случае формировалась дробная часть, и работа, таким образом, окончена. Если JJ = 1, можно заслать точку в IT(N). Теперь J = L2, это означает, что в K не осталось цифр для обработки, и работа окончена.

В противном случае переходим к следующему значению N. Значение JJ должно указывать на (J + 1)-ю цифру K(J первых цифр относились к целой части A), а JK должно указывать на последнюю цифру K, т. е. оно равно L2. Снова возвращаемся в цикл DØ 11, из которого на этот раз выход будет обязательно при JJ ≠ 1.

Посмотрим теперь, что происходит, когда A меньше 1 (целая часть равна нулю). Этот случай рассматривался после выполнения оператора IF (J.GT.0) GØ TØ 10. Наша цель — преобразовать эту дробную часть в целое число K, в разряде единиц которого будет стоять последняя из печатаемых

[1]) В некоторых случаях округление приведет к ошибке. Проанализируйте, например, что произойдет, если A = 0.9999, а LL = 2. — *Прим. ред.*

```
      SUBROUTINE ALIGNE(IT,LL,A)
      DIMENSION IT(1)
      DIMENSION ICH(10)
      DATA ICH/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA IB/1H /,IM/1H-/,IP/1H./,IA/1H*/
      DO 4 I=1,8
4     IT(I)=IB
      IT(2)=IP
      L=LL
      IF(L.LT.3.OR.L.GT.8) L=8
      IF(A.NE.0.) GO TO 2
      DO 3 M=3,L
      IT(M)=ICH(1)
3     CONTINUE
      RETURN
2     CONTINUE
      L2=L-2
      J=ALOG10(ABS(A))+1
      IF(J.GT.L2) GO TO 20
      IF(A.LT.0.) IT(1)=IM
      IF(J.GT.0) GO TO 10
      J=2
      N=3
      JJ=3
      JK=L
      L2=L
      GO TO 1
10    CONTINUE
      N=2
      JJ=1
      JK=J
1     CONTINUE
      K=ABS(A)*10**(L2-J)+0.5
12    CONTINUE
      DO 11 M=JJ,JK
      IC=K/10**(L2-M)
      K=K-IC*10**(L2-M)
      IT(N)=ICH(IC+1 )
      N=N+1
11    CONTINUE
      IF(JJ.NE.1) RETURN
      IT(N)=IP
      IF(J.EQ.L2) RETURN
      JJ=J+1
      N=N+1
      JK=L2
      GO TO 12
20    CONTINUE
      DO 21 I=1,L
21    IT(I)=IA
      RETURN
      END
```

Рис. 124

цифр A. Чтобы воспользоваться общей программой обработки, которую мы только что рассмотрели, параметрам J, N, JJ, JK и L2 нужно присвоить соответствующие начальные значения [1]).

7.3. ЧИСЛА ПО СПИРАЛИ

(*условие задачи на стр.* 67)

Рассмотрим квадрат со стороной 5 и разобьем его в соответствии с рис. 125. Если квадрат имеет сторону четной длины, центрального элемента не существует.

Таким образом, квадрат разбит на группы из 4 строк, имеющих одинаковое число элементов (A, B, C, D) (E, F, G, H), ... При переходе от первой группы ко второй в строке становится меньше на два элемента. В итоге группу инструкций, которые должны

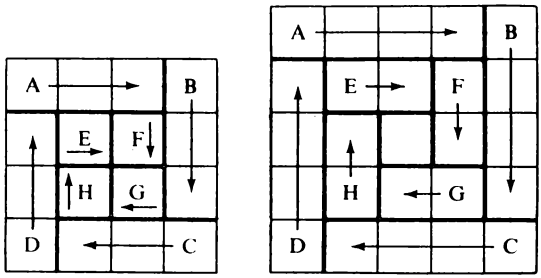


Рис. 125

занести	k-й	элемент	
—	—	—	A
—	—	—	B
—	—	—	C
—	—	—	D

можно объединить в один цикл «с переменной длиной», так как в первый раз он выполняется от 1 до N — 1 (N — длина стороны квадрата), следующий раз от 1 до N — 3, ..., что можно выразить в виде блок-схемы на рис. 126.

Однако задача усложняется тем, что индекс начала каждой строки (A, C) или столбца (B, D) каждый раз меняется, увеличиваясь на единицу для A и B и, наоборот, уменьшаясь на единицу для C и D. Таким образом, было бы более пра-

[1]) J указывает число знаков в целой части A, поэтому J нужно присвоить 0, а не 2, как это сделано в программе. — *Прим. ред.*

вильно написать цикл по K следующим образом:

```
      |  | DØ 2 K = DEB, FIN
      |  | здесь 4 инструкции засылок
    2 |  | CØNTINUE
```

При выходе из этого цикла нужно уменьшить FIN и увеличить DEB на 1, и, пока DEB меньше или равно FIN, цикл

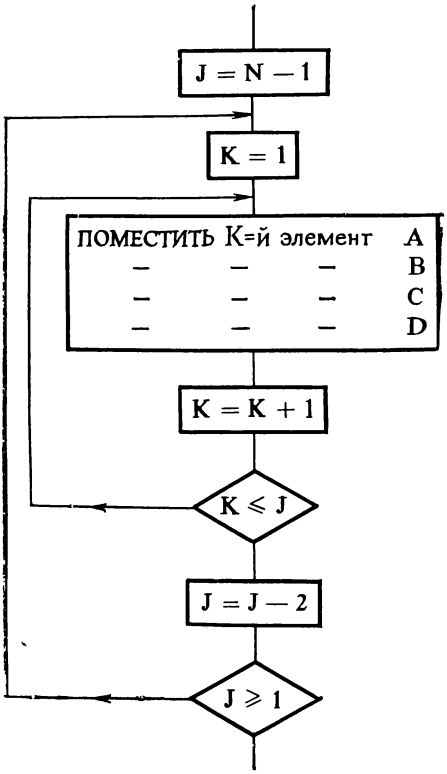


Рис. 126

повторяется. Таким образом, приходим к программе на рис. 127, которую мы прокомментируем более подробно.

Итак, мы будем работать с элементами целого типа и поэтому декларируем их, затем устанавливаем DEB = 1 и FIN = = N — 1, после того как N введено с перфокарты. Переменная NB принимает те значения, которые мы будем располагать по спирали, и вначале оно, естественно, равно 1. M, чис-

ло элементов, обрабатываемых в цикле, очевидно, равно FIN — DEB + 1. Оно представляет собой расстояние между двумя смежными элементами, находящимися в одинаковой

```
C
C          ЧИСЛА ПО СПИРАЛИ  ПРИ 0 < N < 31
C
           INTEGER CARRE(30,30), DEB, FIN
100        READ (5,1,END=101) N
           PRINT 10, N
           DEB=1
           FIN=N-1
           NB=1
4          M=FIN-DEB+1
           DO 2 K=DEB, FIN
           CARRE(DEB,K)=NB
           CARRE(K,FIN+1)=NB+M
           CARRE(FIN+1,FIN-K+DEB+1)=NB+2*M
           CARRE(FIN-K+DEB+1,DEB)=NB+3*M
2          NB=NB+1
           DEB=DEB+1
           FIN=FIN-1
           NB=NB+3*M
           IF (DEB.LE.FIN) GO TO 4
           IF (DEB.EQ.FIN+1) CARRE(DEB,DEB)=NB
C
C          ПЕЧАТЬ РЕЗУЛЬТАТОВ
C
           DO 5 I=1,N
5          PRINT 6, (CARRE(I,J),J=1,N)
           GO TO 100
101        STOP
C
6          FORMAT (/30I4)
10         FORMAT (////10X, ' N =', I3/)
1          FORMAT (I2)
           END
```

Рис. 127

позиции в данный момент. На рис. 128 иллюстрируется этот механизм вычислений.

В цикле обрабатываются по порядку:

1. строка DEB,
2. столбец FIN + 1,
3. строка FIN + 1,
4. столбец DEB.

В первую строку помещается NB, во второй столбец NB + M, поскольку M, как мы только что видели, представляет собой «расстояние» между этими элементами, далее (NB + M) + M или NB + 2 * M и, наконец, NB + 3 * M.

Из рис. 128 и заданного порядка следования инструкций записи при вычислении дополнительных индексов (строки или столбца) получается следующее соответствие:

$$\begin{aligned} C(i_1, j_1) &= NB \\ C(j_1, j_2) &= NB + M \\ C(j_2, j_3) &= NB + 2 * M \\ C(j_3, i_1) &= NB + 3 * M \end{aligned}$$

Изменив значения DEB и FIN, надо увеличить NB. На рис. 128 мы видим, что надо прибавить $3 * M$. Те же операции повторятся после сравнения DEB и FIN, если произойдет переход к инструкции с меткой 4.

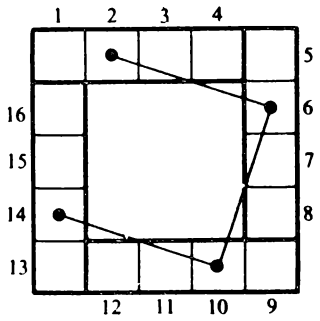


Рис. 128

Но эта проверка недостаточна для квадратов со стороной нечетной длины, имеющих в конце один-единственный центральный элемент. Требуется еще одна проверка для засылки туда последнего значения NB. Эта проверка всегда дает отрицательный ответ, если N четно.

Печать сама по себе не вызывает никаких затруднений, но следует отметить, что необходим неявный цикл, поскольку число элементов, которые надо напечатать, каждый раз равно N, и нельзя написать FØRMAT вида (NI4); поэтому-то и нужно, чтобы началом инструкции печати вызывался переход к новой строке. После печати повторяется ввод новой карты с числом N, и если карт больше нет, то работа завершена.

Решение правильно для $N = 1$, но, очевидно, неверно для $N \leqslant 0$. Чтобы не утяжелять программу, эта проверка опущена.

7.4. ЧИСЛА ПО СПИРАЛИ (КОМПЛЕКСНЫЕ ЧИСЛА ВСЕ УПРОЩАЮТ)

(*условие задачи на стр.* 67)

Сразу оговоримся, что в смысле эффективности это решение не является замечательным, так как оно приблизительно вдвое медленнее предыдущего. Но, поскольку комплексные числа используются редко, мы воспользуемся случаем показать, как это делается.

Как мы уже говорили, координаты клетки находятся из координат предыдущей прибавлением к ним одного из следующих чисел в соответствии с направлением перемещения:

$$(1, 0) \rightarrow \qquad (0, 1) \downarrow \qquad (-1, 0) \leftarrow \qquad (0, -1) \uparrow$$

Эти числа составляют комплексный одномерный массив RØUTE.

На этот раз квадрат будет составлен из «строк» другим способом. На рис. 129 показано это разбиение для $N = 5$ и

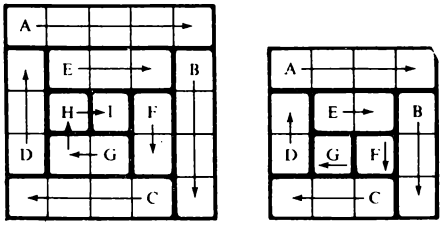


Рис. 129

$N = 4$. Эти случаи уже не различаются в зависимости от четности N, и работа будет продолжаться над множеством пар (строка — столбец), длина которых L уменьшается каждый раз на 1, до тех пор пока L не станет равной нулю.

Первая строка представляет собой особый случай: она состоит из N элементов. При $N = 5$ приходим к последовательности засылок:

××××× 5	×××× 4	×××× 4	××× 3	××× 3	×× 2	×× 2	× 1	× 1

В общем случае имеем

$N + 2(1 + 2 + \ldots)$ элементов, т. е. $N + N(N - 1) = N^2$ элементов.

Каждый раз при переходе от группы длины L к группе длины L — 1 надо изменить направление. Направления же определяются содержимым комплексного массива RØUTE. Таким образом, индекс выбираемого в этом массиве элемента увеличивается на единицу при каждой перемене группы, т. е. индекс I массива RØUTE определяется следующим выражением:

MØD (J, 4) + 1 (рис. 130).

Сформировали отдельно первую строку (цикл DØ 3), полагаем J = 1, тогда индекс I = 2 укажет правильное направление (спуск). M представляет собой индекс элемента массива CARRE, в который засылается следующее число. Только что было занесено N чисел, теперь очередь (N + 1)-го. Всего надо занести L = N — 1 чисел.

Таким образом, приходим к циклу DØ 5 MM = 1, 2, в котором MM служит лишь для того, чтобы повторить два раза внутренний цикл DØ 4 (на самом деле надо занести две группы). Вычисление I производится после DØ 5, так как J изменяется для каждой группы, поскольку мы изменяем направление. Цикл DØ 4 заносит L чисел, причем каждое последующее определяется из предыдущего, как мы уже видели раньше.

Каждый раз, когда кончается совокупность из двух групп, надо переходить к следующей, в которой число элементов L уменьшается на 1. Так будет продолжаться до тех пор, пока N не станет равным нулю.

Теперь переходим к печати. Речь идет не о печати просто пар координат, а чисел, расположенных по спирали. Для этого используем одномерный массив SØRTIE типа INTEGER, в который поместим

- в первый раз все комплексные числа, у которых мнимая часть равна 1, расположенные в порядке возрастания действительных частей: (1, 1), (2, 1), (3, 1), ...,
- во второй раз все те, у которых мнимая часть равна 2, расположенные также в порядке возрастания действительных частей: (1, 2), (2, 2), (3, 2), ...

и т. д.

На самом деле мы помещаем в массив типа INTEGER не *комплексные* числа (непонятно, как бы мы это могли сделать!), а *индексы* элементов в массиве CARRE, где эти числа находятся.

Надо напечатать N строк. В цикле DØ 7 повторяется N раз инструкция PRINT и предшествующий ей цикл DØ 8, работающий с N2 членами массива CARRE. Заметим, что операция N2 = N * N вынесена из этих циклов.

В этом цикле DØ 8 мы ищем все числа, мнимая часть которых равна I, что представляет собой порядковый номер строки, которую надо печатать. Когда найдено одно такое

```
C       ЧИСЛА ПО СПИРАЛИ - РЕШЕНИЕ В КОМПЛЕКСНОМ ВИДЕ
C       ДЕЙСТВИТЕЛЬНО ОТ N= 1 ДО N= 30, ВКЛЮЧАЯ ГРАНИЦЫ
C
        INTEGER SORTIE (30)
        COMPLEX ROUTE(4), CARRE(900)
        DATA ROUTE /(0.,1.),(1.,0.),(0.,-1.),(-1.,0.)/
C
11      READ (5,1,END=12) N
        PRINT 2, N
C
C
C       ВЫЧИСЛЕНИЕ ВЕРХНЕЙ СТРОКИ МАССИВА CARRE
        DO 3 I=1,N
3       CARRE(I)=CMPLX(1.,FLOAT(I))
        J=1
        M=N+1
        L=N-1
5       DO 5 MM=1,2
        I=MOD(J,4)+1
        DO 4 NN=1,L
        CARRE(M)=CARRE(M-1)+ROUTE(I)
4       M=M+1
5       J=J+1
        L=L-1
        IF (L.GT.0) GO TO 6
C
C       ПЕЧАТЬ РЕЗУЛЬТАТОВ
        N2=N*N
        DO 7 I=1,N
        DO 8 L=1,N2
        IF (IFIX(REAL(CARRE(L))).NE.I) GO TO 8
        J=AIMAG(CARRE(L))
        SORTIE(J)=L
8       CONTINUE
7       PRINT 9, (SORTIE(J),J=1,N)
        GO TO 11
12      STOP
1       FORMAT (I2)
2       FORMAT (///10X'N = ',I2//)
9       FORMAT (/30I4)
        END
```

Рис. 130

число, его действительная часть показывает, какую ячейку должен занимать элемент в строке:

$$J = \text{REAL}(\text{CARRE}(L)).$$

Таким образом, надо заслать индекс CARRE, т. е. L, в массив SØRTIE(J). Не надо никаких начальных установок, потому что каждый раз мы уверены, что найдутся N элементов, ко-

торые надо заслать. В конце цикла DØ7 снова вводим значение N. Если его нет, попадаем на инструкцию STØP.

Заметим, что для того, чтобы сравнить мнимую часть комплексного числа с целым, надо использовать две функции: AIMAG и IFIX. В операторе DATA, определяющем массив RØUTE, задаются пары (re, im), элементами которых являются вещественные числа.

7.5. СНАРУЖИ ИЛИ ВНУТРИ?

(*условие задачи на стр.* 68)

Вначале вводим с карт координаты 20 вершин, которые отперфорированы парами по 4 колонки на значение (рис. 131). Избегайте такой ошибки:

```
      READ 1, X, Y
или   READ 1, (X(I), I = 1, 20), (Y(I), I = 1, 20),
```

так как и в том и в другом случае будет считаться, что на первой карте отперфорированы значения X, а на второй — значения Y.

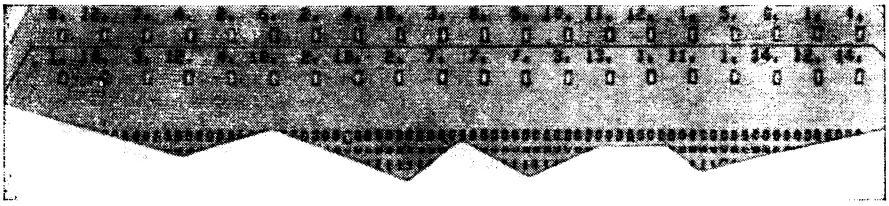


Рис. 131

Для печати используется неявный DØ-цикл, такой же по виду, причем значение индекса I служит также для печати номера вершины. Это довольно интересная возможность, на которую часто не обращают внимания. Затем вводятся XP и YP, координаты точки P и начинается вычисление угла между последней и первой точкой; причину этого мы поймем в дальнейшем. Для этого используется функция ATAN2, так как легко находится тангенс этого угла (рис. 132).

Имеем

$$\varphi = \beta - \alpha$$

и

$$\operatorname{tg}\beta = \frac{Y_1 - Y_p}{X_1 - X_p}, \quad \operatorname{tg}\alpha = \frac{Y_{20} - Y_p}{X_{20} - X_p},$$

откуда

$$\operatorname{Arctg}\beta - \operatorname{Arctg}\alpha = \varphi.$$

Если точки n, n + 1, n + 2, ... соответствуют одинаковому направлению вращения, все будет происходить правильно. Но если точка n + 3 расположена *сзади* точки n + 2, вычисленное значение Arctg будет неверно, в чем можно убедиться, глядя на рис. 132.

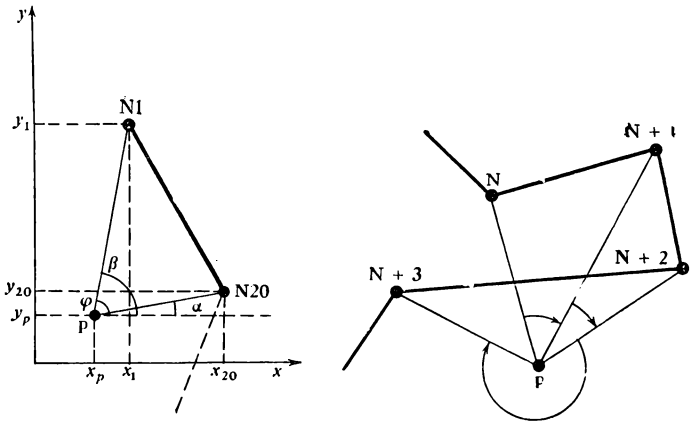


Рис. 132

Таким образом, его надо исправить, прибавив к нему или отняв от него 2π, в зависимости от знака найденного угла. Поэтому просто проверим с помощью оператора IF(ABS(PHI) ..., не было ли сделано больше половины оборота (две точки, лежащие на одной прямой, нельзя увидеть под углом, большим π радиан). Если результат проверки положителен, в силу направления вращения, выбранного в программе, надо отнять 2π, если PHI положителен, и прибавить 2π в противном случае.

Частное от деления ABS(PHI)/PHI равно +1 в первом случае и —1 во втором, и после умножения на величину 2π (6.283185) получается искомая поправка. Можно было бы также написать

IF(...)PHI = PHI — SIGN(6.283185, PHI);

при этом функция SIGN дает значение, равное абсолютному значению первого аргумента (здесь 2π) со знаком второго (рис. 133).

Затем вычислим значения углов для всех I от 1 до 20. Поскольку каждый раз нам нужны последовательные значения

```
C       С Н А Р У Ж И   И Л И   В Н У Т Р И ?
C
        DIMENSION X(20), Y(20)
        READ 1, (X(I), Y(I), I=1,20)
        PRINT 3, (I,X(I),Y(I), I=1,20)
C
7       READ (5,1,END=2) XP, YP
        PHI=ATAN2(Y(1)-YP,X(1)-XP)-ATAN2(Y(20)-YP,X(20)-XP)
        IF (ABS(PHI).GT.3.141593)PHI=PHI-6.283185*ABS(PHI)/PHI
        DO 4 I=1,19
        DPHI=ATAN2(Y(I+1)-YP,X(I+1)-XP)-ATAN2(Y(I)-YP,X(I)-XP)
        IF (ABS(DPHI).GT.3.141593) DPHI=DPHI-6.283185*ABS(DPHI)/DPH
4       PHI=PHI+DPHI
        IF (ABS(PHI).LT.0001) PRINT 6, XP, YP
        IF (ABS(PHI).GT.6.28) PRINT 5, XP, YP
        GO TO 7
2       STOP
1       FORMAT (20F4.0)
3       FORMAT (30X,9H МНОГОУГ.:) //4(I3,2F5.0,5X))
5       FORMAT (//10X12H ТОЧКА     X = F5.0,7H  И  Y = F5.0,
     1  18H ЛЕЖИТ ВНУТРИ/),
6       FORMAT (//10X 12HТОЧКА      X = F5.0, 7H  И  Y = F5.0,
     1  18H ЛЕЖИТ СНАРУЖИ/)
        END
```

МНОГОУГ.:

```
  1   9.   12.      2   7.   4.      3   2.   6.      4   2.   4.
  5  10.    3.      6   8.   5.      7  10.  11.      8  12.   1.
  9   5.    0.     10   1.   4.     11   1.  10.     12   3.  12.
 13   4.   10.     14   2.  10.     15   2.   7.     16   7.   7.
 17   3.   13.     18   1.  11.     19   1.  14.     20  12.  14.
```

```
        ТОЧКА     X =   7. ET Y =  11. ЛЕЖИТ ВНУТРИ

        ТОЧКА     X =   4. ET Y =   8. ЛЕЖИТ СНАРУЖИ

        ТОЧКА     X =   5. ET Y =   4. ЛЕЖИТ СНАРУЖИ

        ТОЧКА     X =   2. ET Y =   1. ЛЕЖИТ СНАРУЖИ

        ТОЧКА     X = -10. ET Y =  -2. ЛЕЖИТ СНАРУЖИ

        ТОЧКА     X =   5. ET Y =   5. ЛЕЖИТ ВНУТРИ

        ТОЧКА     X =   0. ET Y =   0. ЛЕЖИТ СНАРУЖИ
```

Рис. 133

X и Y для вычисления угла, начинаем с того, что перенесем первые значения координат в 21-ю искусственную точку: Y(21), X(21). Таким образом, мы избегаем необходимости по отдельности вычислять угол, заключенный между первой и

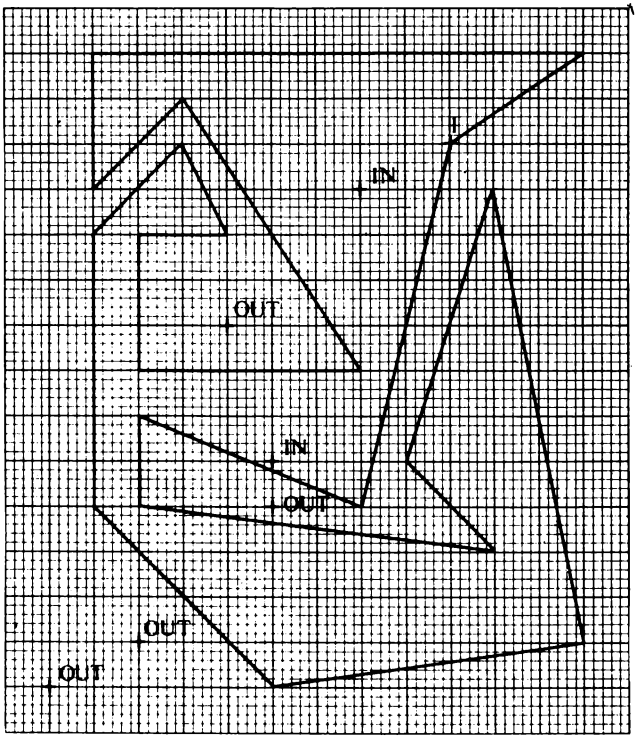


Рис. 134

последней точкой. На каждом шаге вычисленные углы суммируются.

При выходе из цикла PHI равно 0 или 2π (с точностью до знака). В самом деле, не следует забывать, что знак результата зависит от принятого направления вращения и от порядка следования определяемых вершин многоугольника.

Прежде чем приступать к сравнению, следует подумать о том, что надо учесть ошибки округления, возникшие в ходе вычисления. Поэтому вместо того, чтобы при проверке сравнивать с 0, надо использовать 0, 0001. Поскольку, кроме того, все отрицательные числа меньше положительного числа, как

бы ни было оно мало, и неизвестен знак ошибки при вычислении PHI, лучше рассматривать абсолютное значение PHI, а не само PHI.

Если PHI отлично от нуля, это означает, что точка находится внутри; в противном случае она находится снаружи и поскольку в Фортране нет инструкции типа IF(...) THEN... ELSE ..., надо использовать два логических оператора IF (можно было бы использовать всего один, но тогда понадобилось бы два GØ TØ, что неизящно).

На чертеже (рис. 134), полученном с помощью плоттера Бенсона, графически представлены результаты рис. 133. Начало координат — это точка, отмеченная маркером «OUT» в левом нижнем углу.

В заключение заметим, что трудности, с которыми мы столкнулись при решении этой задачи, носят прежде всего математический характер, при этом проблемы, связанные с программированием, остаются на элементарном уровне.

7.6. ЮЛИЙ, ГРИГОРИЙ И ИХ КАЛЕНДАРИ

(*условие задачи на стр.* 69)

Определяем массив с именем MØIS, состоящий из 12 чисел, каждое из которых представляет собой количество дней, прошедших с 1 января текущего года до (но не включая)

```
C       ЮЛИАНСКАЯ ДАТА ПО ГРИГОРИАНСКОМУ КАЛЕНДАРЮ
C
      INTEGER FUNCTICN DATE (A, M, J)
      INTEGER A
      DIMENSION MOIS(12)
      DATA MOIS /0,31.,59.,90.,120.,151.,181.,212.,243.,273.,304.,334./
      DATE=J+MOIS(M)+1721060+365*A+A/4-A/100+A/400
      IF((MOD(A,4).EQ.0.AND.MOD(A,100).NE.0.OR.MOD(A,400).EQ.0)
     1   .AND.M.LT.3) DATE=DATE-1
      RETURN
      END
```

Рис. 135

1 числа месяца, данного в аргументе; это для невисокосных годов. Как можно видеть в программе на рис. 135, J + + MØIS(M) дает, таким образом, номер дня месяца M в году.

Чтобы узнать количество дней, соответствующих A прошедшим годам, достаточно, таким образом, умножить A на 365, а затем добавить столько дней, сколько было високосных лет:

A/4 соответствует високосному году каждые 4 года,

A/100 дает поправку этой величины для вековых годов,

A/400 дает поправку предыдущей поправки для високосных годов каждые 400 лет.

Затем прибавляется 1 721 060, чтобы получить абсолютное значение (в смысле, противоположном «относительному») DATE, поскольку начало григорианского календаря приходится на день с номером 1 721 060 по юлианскому календарю.

Остается внести поправку, если данный год A сам является високосным и месяц M позже февраля. В этом случае достаточно прибавить 1 к DATE. Если месяц был январь или февраль, то ничего не надо исправлять, так как поправка приходится на конец февраля — начало марта.

В. Кларк и П. Стумпф из Национальной радиоастрономической обсерватории (США) составили более изящную программу, которая представлена на рис. 136. Интересно пояснить ее функционирование:

```
      REAL FUNCTION DJL*8(IY,IM,ID)
C     NRAO 5/1 F(D) DJL F4S LCD=5
      JDN=(IM*3057)/100+ID+((5-IM/3)/5)*(2-(4-MOD(IY,4))/4+
     1   (100-MOD(IY,100))/100-(400-MOD(IY,400))/400)+1721028
     2   +(IY*365+IY/4-IY/100+IY/400)
      DJL=DFLOAT(JDN)-0.5D0
      RETURN
      END
```

Рис. 136

Разложим запись программы (а в сущности инструкции JDN = = ..., за исключением карты-продолжения с номером 2) следующим образом:

$$\begin{cases} N1 = \dfrac{IM * 3057}{100} - 32 \\ N2 = \dfrac{4 - MOD(IY, 4)}{4} + \dfrac{100 - MOD(IY, 100)}{100} + \\ \qquad\qquad + \dfrac{400 - MOD(IY, 400)}{400} \\ N3 = \dfrac{5 - \dfrac{IM}{3}}{5} \\ N4 = N1 + N3(2 - N2) \end{cases}$$

N2 равно 1, если год високосный (тот же принцип, что и в первом решении).

N3 равно 1 для января и февраля, 0 для других месяцев (IM/3 равно 0 только в первом случае).

N1 немного сложнее. Она является эквивалентом переменной MØIS (M) из предыдущего решения, за исключением января и февраля:

IM	1	2	3	4	5	6	7	8	9	10	11	12
N1	– 2	29	59	90	120	151	181	212	243	273	304	334

Этот таинственный коэффициент 3057 легко объясняется, если заметить, что выражение записывается:

$$\frac{3057 * IM}{100},$$

что равносильно вычислению 30,57 * IM, т. е. умножается номер месяца на число, немного большее 30 и немного меньшее 31 (число дней в обычных месяцах). Формула остается верной, если взять 3056 или 3058, и понятно, что 3057 взято для «безопасности»!

Чтобы сделать поправку, надо отнять 2 для каждого нормального года и 1 для високосных, если речь идет о январе или феврале. Имеем

	Январь	Февраль	Другие месяцы
Високосные годы	—1	30	без изм
Обычные годы	0	31	без изм

Это еще не совсем точно, но поправка для високосных годов:

$$\frac{IY}{4} - \frac{IY}{100} + \frac{IY}{400}$$ (карта-продолжение с номером 2)

делается без учета месяца, что даст 0 и 31 для первых двух месяцев и увеличит на 1 число дней в следующих месяцах.

Константа приведения к началу календаря, 1721060, здесь уменьшена до 1721028, чтобы учесть число «32». Прибавляют 0,5 к найденному целому, переведенному в двойную точность, чтобы можно было учитывать даже час, если понадобится (выраженный в дробной части от суток). При этом предполагается, что результат, DJL, дается в полдень. Оставшаяся часть формулы не требует комментариев.

День недели

Так же как мы это обычно делаем, ограничим число литер в слове четырьмя. К счастью, ни один день недели не имеет

в своем названии больше 8 букв[1]); таким образом, достаточно будет массива из 14 слов, чтобы разместить все их названия. Вычисляется раз и навсегда «юлианский» номер 1 января 1583 г. и из него вычитается число, заключенное между 0 и 6, что позволит совместить начало календаря с правильным днем недели. Мы не знаем априори, какой это день, попробуем уточнить его экспериментально; как это делается, мы увидим в дальнейшем.

Затем вводится карта данных (рис. 137) и вычисляется разность между юлианским номером соответствующего дня

```
C       НАЙТИ ДЕНЬ НЕДЕЛИ ДЛЯ ЗАДАННОЙ ДАТЫ
C
        INTEGER DATE
        DIMENSION JOUR (14)
        DATA JOUR/4HLUND,1HI,4HMARD,1HI,4HMERC,4HREDI,4HJEUD,1HI,
     1             4HVEND,4HREDI,4HSAME,2HDI,4HDIMA,4HNCHE/
C
        K=DATE(1583,1,1)-5
4       READ (5,1,END=2) J, M, IA
1       FORMAT (2I2,I4)
        N=DATE(IA,M,J)-K
        IF (N.GE.0) GO TO 3
        PRINT 5, J, M, IA
5       FORMAT (1HO,3I5,20H: DATE TROP ANCIENNE!
        GO TO 4
3       I=(MOD(N,7)+1)*2
        PRINT 6, J, M, IA, JOUR(I-1), JOUR(I)
6       FORMAT (3HCLE,2I3,I5,1OH ETAIT UN , 2A4)
        GO TO 4
2       STOP
        END
```

Рис. 137

и вычисленным заранее юлианским номером начала календаря. Допустим, что с точностью до дня недели 1 января 1583 г. является правильным началом. Если разность отрицательна, печатается сообщение об ошибке, в противном случае вычисляется остаток от деления этой разности на 7, что дает, если считать от 0 до 6, номер дня недели (0 — понедельник, 1 — вторник, ..., 6 — воскресенье). Добавим единицу, чтобы можно было использовать массив JØUR, и умножим на 2, так как на каждый день недели отводятся 2 слова памяти.

Остается напечатать ... и уточнить значение K!

Полагаем наугад K = (DATE (1583, 1, 1) — 3 и вводим карту с какой-нибудь недавней датой, например того дня, когда происходят вычисления. Констатируем отставание в 2 дня. Таким образом, правильное значение константы равно 5. Значит, 1 января 1583 г. приходилось на субботу.

[1]) Во французском языке. — *Прим. перев.*

Орфографическое замечание: если нам нужна дата в будущем, вместо «БЫЛО» надо печатать «БУДЕТ». Если в используемой вычислительной машине можно вызвать нестандартную подпрограмму, которая сообщит сегодняшнюю дату, то нетрудно исправить программу в этом смысле. Если нет, достаточно заменить «БЫЛО» на «=» либо на «:».

7.7. ТРИ ПРОГРАММЫ, ПОЛНЫЕ ОШИБОК, КОТОРЫЕ НАДО НАЙТИ ДО КОМПИЛЯТОРА

(*условие задачи на стр.* 70)

Первая программа

I Инстр. 1 — DIMENSION написано с 0 (нулем) вместо O (Ø)

I Инстр. 2 — имеется лишняя запятая после указания метки в инструкции FØRMAT (1)

I Инстр. 4 — аргументом функции ALØG10 не может быть число типа INTEGER. Надо было бы написать «23.» либо FLØAT (23)

II Инстр. 5 — переменную I нельзя использовать в DØ-цикле, поскольку она описана как массив в инструкции 1

— J — 1 — выражение, что запрещено в этом месте DØ-цикла

I Инстр. 7 — здесь нельзя использовать FØRMAT с меткой 10, так как в нем нет никакой спецификации (тип E или F) для обработки переменной Q

II Инстр. 8 — нельзя сравнивать целое число (I(1)) с вещественным (.5)

— метка 3 не существует

II Инстр. 10 — A — массив и должен быть записан с двумя индексами

— то же самое и для I, но с одним индексом

I Инстр. 12 — та же ошибка, что в инструкции 10

I Инстр. 13 — для печати вещественных чисел используется спецификация, предназначенная для целых чисел (I5)

I Инстр. 14 — лишний знак «=»

III Инстр. 15 — A(4, 6) не существует

— лишняя правая скобка

— 1 это метка формата

Вторая программа

I Инстр. 1 — написано F0RMAT вместо FØRMAT

I Инстр. 2 — массив L не описан, поэтому DATA содержит 5 лишних констант: 5 последних

III Инстр. 4 — инструкция с описанием массива L должна была предшествовать DATA
— нельзя задавать размер массива с помощью переменной (P(M))
— метка (1) не имеет смысла при декларации

I Инстр. 5 — не хватает запятой между 5 и A

II Инстр. 8 — лишняя запятая между 2 и J
— L(2) неправильно, так как в качестве параметра DØ-цикла нельзя использовать переменную с индексом

III Инстр. 9 — размерность A не была описана. Поскольку имя A уже фигурировало без индекса в инструкции READ, оно рассматривается как простая переменная. В противном случае компилятор мог бы предположить, что речь идет о функции с именем A
— после J + 1 не хватает правой скобки
— 3 является меткой формата. Поэтому нельзя к ней отсылать при помощи GØ TØ

I Инстр. 15 — не присвоено начальное значение Z. Поэтому его содержимое не определено, и в момент выполнения наверняка будет ошибка

I Инстр. 16 — в инструкции 4 массив X описан как двумерный; здесь же не хватает одного индекса

I Инстр. 17 — в инструкции 4 массив K описан как двумерный, поэтому его нельзя использовать как простую переменную

II Инстр. 18 — лишнее R в слове PRRINT
— 2 не является меткой формата

I Инстр. 20 — метка 83 уже определена (в инструкции 16)

I Инстр. 21 — метки 27 не существует

I Инстр. 24 — в инструкции 4 массив L описан как одномерный, поэтому его нельзя использовать как простую переменную

II Инстр. 25 — обычно не разрешено использовать L/2 в индексном выражении
— лишняя запятая между PRINT и 18

II Инстр. 26 — между 2E15 и 8 вместо точки стоит запятая
— не хватает запятой между F5.0 и F10.1

Можно было также отметить, что значение M в инструкции 21 равно 0, если судить по инструкциям 19 и 20, в то время как индексы в Фортране начинаются с 1.

Третья программа

Здесь собрано довольно большое количество ложных ошибок, которые будут разобраны после настоящих

IIII Инстр. 1 — DIMENSSIØN написано с двумя «S»
— не хватает скобки после B(15)
— между 4 и 7 в «C» стоит точка вместо запятой
— список декларируемых переменных не должен заключаться в скобки

I Инстр. 4 — в CØMMØN произошло «короткое замыкание». В силу EQUIVALENCE (A(1), B(1)) и EQUIVALENCE (X, B(10)) B(10) и A(7) должны оказаться в одной и той же ячейке. С другой стороны, A(10) отделяется от A(7) двумя словами: A(8) и A(9). Таким образом, CØMMØN «раздирают на части», так как B(10) (и, следовательно, A(7)) должны совместиться с X:

$$\left.\begin{array}{l} A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_7\ \ A_8\ \ A_9\ \ A_{10} \\ B_4\ B_5\ B_6\ B_7\ B_8\ B_9\ B_{10}\ B_{11}\ B_{12}\ B_{13} \\ \qquad\qquad\qquad\quad X\qquad\quad YZ \end{array}\right\} \text{EQUIVALENCE}$$

$$X\ YZ \qquad \text{CØMMØN}$$

I Инстр. 5 — I присвоено некорректное начальное значение. Надо было написать (175), а не (175.)

I Инстр. 10 — TRUC — логическая переменная. Следовательно, квадратный корень в применении к ней не имеет смысла

I Инстр. 11 — метки 12 не существует

II Инстр. 12 — аргументом функции ALØG10 не должно быть целочисленное выражение
— инструкция должна быть помечена, иначе она никогда не будет выполнена, поскольку следует за арифметической инструкцией IF

Инстр. 14 — запрещается писать выражение (A(J) — 1) слева от знака «=»

II Инстр. 17 — не хватает левой скобки
— у переменной A должен быть один индекс

I Инстр. 18 — имя «CØSINUS» состоит из 7 литер; одна лишняя

I Инстр. 20 — RETURN не имеет смысла в программе, которая не является ни функцией (FUNCTIØN), ни подпрограммой (SUBRØUTINE),

Ложные ошибки

Инстр. 9 — DØ3 I = 1 1 0 не есть инструкция DØ, но присваивание переменной DØ3I значения 110. Мы признаем, что злонамеренно поместили пробелы, чтобы ввести читателя в заблуждение, и дополнили иллюзию инструкцией CØNTINUE с меткой 3

Инстр. 11 — часто ошибаются, думая, что надо писать IF(Z — 0) в арифметическом IF или IF (A.EQ..TRUE.) в логическом IF

Инстр. 15 — в Фортране нет стандартных имен. Поэтому мы имеем право назвать переменную «IF»

Инстр. 16 — то же замечание применимо к имени GØ TØ, которое здесь обозначает переменную

Инстр. 19 — STØP здесь переменная и, кроме того, READ(10) вполне может быть вызовом функции READ (предполагается, что эта функция написана на Фортране программистом), которой в качестве аргумента передается число 10

8. НЕСКОЛЬКО НЕЧИСЛОВЫХ ЗАДАЧ

8.1. ЗАДАЧА О ШАРИКЕ И ЗАСЛОНКАХ

(*условие задачи на стр.* 72)

Поскольку заслонка может принимать всего два положения, естественно выбрать для ее представления логическую переменную. Так как имеется 31 заслонка, опишем массив, состоящий из 31 логической переменной: CLAPET (31).

Изучая устройство с заслонками, замечаем, что заслонка с номером n может выбросить шарик либо к заслонке с номером 2n, либо к заслонке с номером 2n + 1. Перемена положения заслонки записывается выражением вида:

$$\text{CLAPET (I)} = .\text{NØT.CLAPET (I)}.$$

Условимся, что если заслонка направляет шарик влево, значение этой переменной будет равно .TRUE., а если она повернется вправо, то значение будет равно .FALSE.. Таким образом, положение заслонок первоначально определяется картой, на которой в первой 31 колонке перфорируется T или F в зависимости от ориентации заслонки, имеющей соответствующий номер (рис. 138).

Итак, вначале бросают шарик в воронку № 1 (N = 1), затем он пройдет сквозь 4 ряда заслонок, 5-й ряд составляют 32 приемные ячейки (от 32 до 63). Как мы видим на рис. 139,

Рис. 138

пять последовательных падений шарика обрабатываются в цикле DØ 2. На каждом проходе цикла вычисляется новое значение N, а затем переход соответствующей заслонки в другое положение. При выходе из цикла значение N представляет собой номер достигнутой приемной ячейки.

```
      LOGICAL   CLAPET(31),BLANC, ASTER, GAUCHE,DROITE
      DATA BLANC/1H /,ASTER/1H*/,GAUCHE/1H//,DROITE/1H\/
      J=0
      READ 5, CLAPET
1     N=1
      DO 2 I=1,5
      M=2*N+1
      IF (CLAPET(N)) M=M-1
      CLAPET(N)=.NOT.CLAPET(N)
2     N=M
      J=J+1
      IF(N.NE.41) GO TO 1
      PRINT 6, J
      DO 7 I=1,31
      IF (CLAPET(I)) GO TO 8
      CLAPET(I)=DROITE
      GO TO 7
8     CLAPET(I)=GAUCHE
7     CONTINUE
      K=2*N-64
      PRINT 3, CLAPET, (BLANC,I=1,K),ASTER
3     FORMAT (32X,A1///,16X,A1,31X,A1///8X,A1,3(15X,A1)///
     1  4X,A1,7(7X,A1)///2X,A1,15(3X,A1)/1X16(4H!!! )/1H*,64A1)
5     FORMAT (31L1)
6     FORMAT ('ВЫИГРЫШ ПОСЛЕ',I3, 8H ПАРТИЙ //)
      STOP
      END
```

Рис. 139

Чтобы получить M, новое значение переменной N, произвольно предполагается, что заслонка повернулась вправо и, следовательно, M = 2 * N + 1. Если она повернулась влево, надо исправить найденное значение, т. е. вычесть из него еди-

ницу: M = M — 1. Затем можно заставить заслонку поменять положение (.NØT.CLAPET(N)).

Ни в коем случае не следует писать непосредственно N = 2 * N + 1, потому что тогда проверка IF(CLAPET(N)) не имела бы смысла, так как значение N уже изменилось. Вспомогательная переменная M позволяет выйти из положения.

Параметр J служит для счета «партий».

Если N при выходе из цикла не равно 41, то шарик снова вводится в игру (в воронку № 1). Когда «партия выиграна»,

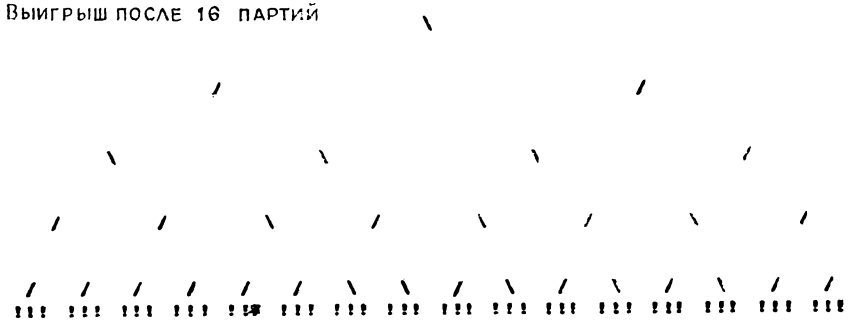


Рис. 140

надо приступать к печати. Для начала в массиве CLAPET логические значения заменяются литерами / или \, которые были описаны под именами DRØITE и GAUCHE как логические переменные и которым в качестве начальных значений в инструкции DATA были присвоены «литерные» константы.

Еще раз отмечаем полезность инструкции DATA, которая позволяет переменной без преобразования присвоить информацию иного типа, чем та, которую она обычно должна содержать.

Сама по себе печать очень проста, только формат следует писать с большой аккуратностью. Переменная K характеризует положение шарика в зависимости от порядкового номера ячейки (N) и от общего состояния игры. Литера + позволяет напечатать строку с наложением * на / в соответствующей ячейке.

ВОПРОСЫ:

1. Почему на рис. 140 звездочка напечатана не внутри ячейки, а рядом?
2. Почему печать результатов показывает положение заслонок, не соответствующее положению шарика?

ОТВЕТЫ:

1. Потому что в печатающем устройстве нет «половинной позиции».
2. В условии задачи требовалось, чтобы положение заслонок было напечатано в момент, когда партия будет выиграна. Поскольку проход шарика сопровождается переменой положения заслонок, естественно, что то положение, которое было до прохождения шарика, сменилось на противоположное после его прохождения.

8.2. ВОСЕМЬ ФЕРЗЕЙ НА ШАХМАТНОЙ ДОСКЕ

(*условие задачи на стр.* 73)

Пусть I1, I2, ..., I8 — переменные, представляющие соответственно положение первого ферзя, второго ферзя, ..., восьмого ферзя. На самом деле казалось бы удобнее

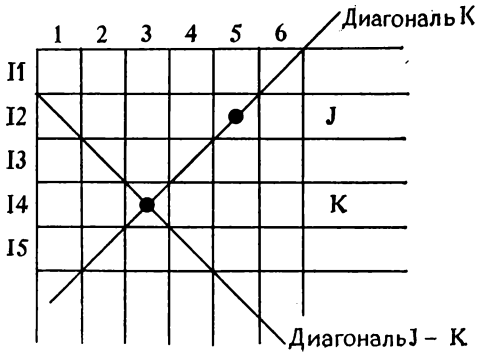


Рис. 141

решать задачу, имея массив из 8 элементов I(8). Но это привело бы нас к написанию инструкций такого вида: DO1 I(n) = 1, 8, что строго воспрещается. Поэтому мы воспользуемся инструкциями EQUIVALENCE и CØMMØN. Последняя позволит нам к тому же обратиться к функции TEST, полезность которой в данной программе обнаружится позже.

Каждая из переменных I1, I2, ..., I8 будет принимать все значения 1, 2, ..., 8, представляющие положение соответствующего ферзя. Два ферзя J и K считаются «под боем», если они находятся на одной и той же вертикали ($I_j = I_k$) или на одной и той же диагонали ($I_j = I_k + k - j$ либо $I_j = I_k + j - k$). Этот случай представлен на рис. 141.

Начиная с DØ-цикла, определяющего I2, и вплоть до цикла, определяющего I8 (и включая его), будет произво-

диться проверка этих двух условий для каждого нового значения I2, I3, ..., I8, и если одно из них удовлетворяется, это будет означать, что проверяемое значение не подходит и надо попробовать следующее значение.

Можно было бы написать

```
DØ 1 I1 = 1, 8
DØ 1 I2 = 1, 8
IF (I2 .EQ. I1 .ØR. I2 .EQ. I1 + 1 .ØR. I2 .EQ. I1 – 1) GØ TØ 1
DØ 1 I3 = 1, 8
IF (I3 .EQ. I2 .ØR. I3 .EQ. I2 + 1 .ØR. I3 .EQ. I2 – 1) GØ TØ 1
IF (I3 .EQ. I1 .ØR. I3 .EQ. I1 + 2 .ØR. I3 .EQ. I1 – 2) GØ TØ 1
DØ 1 I4 = 1, 8
IF (I4 .EQ. I3 ...
    .....
```

что было бы длинно и мучительно. Но благодаря инструкции EQUIVALENCE мы устанавливаем соответствие между I1 и I(1); I2 и I(2); ... и т. д. Кроме того, мы видим, что

для I2 требуется 1 проверка,

для I3 требуются 2 проверки,

для I4 требуются 3 проверки ... и т. д.

Это наводит на мысль поместить эти проверки в DØ-цикл, у которого начальные и конечные значения параметров были бы сначала 1 и 1, затем 1 и 2, ... И вообще после DØ I_n эти значения были бы 1 и (n — 1).

Поскольку общая структура цикла одинакова, вместо того, чтобы каждый раз повторять этот цикл, можно упростить запись (ценой небольшой потери времени), составив подпрограмму типа FUNCTIØN. Было бы чрезвычайно удобно написать

```
DØ 1 I1 = 1, 8
DØ 1 I2 = 1, 8
IF (TEST(...)) GØ TØ 1
DØ 1 I3 = 1, 8
IF (TEST(...)) GØ TØ 1
DØ 1 I4 = 1, 8
IF (TEST(...)) GØ TØ 1
    ......
```

Таким образом, TEST будет *логической* функцией, и мы должны декларировать имя TEST как таковое в главной программе.

Аргументом этой функции будет просто целое число, равное числу операторов IF, которые надо выполнить, т. е., как мы только что видели, 1, затем 2, 3, ..., 7 (рис. 142).

```
C       ВОСЕМЬ ФЕРЗЕЙ НА ШАХМАТНОЙ ДОСКЕ
C
        DIMENSION I(8), K(8)
        COMMON I1,I2,I3,I4,I5,I6,I7,I8
        EQUIVALENCE (I1,I(1))
        LOGICAL TEST
        DATA K1/1HO/,K2/1H*/
        N=0
C
        DO 1 I1=1,8
        DO 2 I2=1,8
        IF (TEST(2)) GO TO 2
        DO 3 I3=1,8
        IF (TEST(3)) GO TO 3
        DO 4 I4=1,8
        IF (TEST(4)) GO TO 4
        DO 5 I5=1,8
        IF (TEST(5)) GO TO 5
        DO 6 I6=1,8
        IF (TEST(6)) GO TO 6
        DO 7 I7=1,8
        IF (TEST(7)) GO TO 7
        DO 8 I8=1,8
        IF (TEST(8)) GO TO 8
        N=N+1
        PRINT 50, N
        DO 40 M=1,8
        DO 30 J=1,8
30      K(J)=K1
        L=I(M)
        K(L)=K2
40      PRINT 20, K
8       CONTINUE
7       CONTINUE
6       CONTINUE
5       CONTINUE
4       CONTINUE
3       CONTINUE
2       CONTINUE
1       CONTINUE
        STOP
20      FORMAT (1X, 8A2)
50      FORMAT (//' РЕШЕНИЕ  N  ', I3/)
        END


        LOGICAL FUNCTION TEST(I)
        COMMON J(8)
        TEST=.TRUE.
        I1=I-1
        DO 1 K=1,I1
        IF (J(I).EQ.J(I-K)-K.OR.J(I).EQ.J(I-K)+K.OR.J(I).EQ.J(K)) RETURN
1       CONTINUE
        TEST=.FALSE.
        RETURN
        END
```

```
РЕШЕНИЕ  N .  1
* O O O O O O O
O O O O * O O O
O O O O O O O *
O O O O O * O O
O O * O O O O O
O O O O O O * O
O * O O O O O O
O O O * O O O O

РЕШЕНИЕ  N   2
* O O O O O O O
O O O O O * O O
O O O O O O O *
O O * O O O O O
O O O O O O * O
O O O * O O O O
O * O O O O O O
O O O O * O O O

РЕШЕНИЕ  N   52
O O O O * O O O
O * O O O O O O
O O O O O * O O
* O O O O O O O
O O O O O O * O
O O O O * O O O
O O O * O O O O
O O * O O O O O

РЕШЕНИЕ  N   53
O O O O * O O O
O * O O O O O O
O O O O O O O *
* O O O O O O O
O O O * O O O O
O O O O O O * O
O O * O O O O O
O O O O O * O O

РЕШЕНИЕ  N   91
O O O O O O O *
O O * O O O O O
* O O O O O O O
O O O O O * O O
O * O O O O O O
O O O O * O O O
O O O O O O * O
O O O * O O O O

РЕШЕНИЕ  N   92
O O O O O O O *
O O O * O O O O
* O O O O O O O
O O * O O O O O
O O O O O O * O
O * O O O O O O
O O O O O O O *
O O O O * O O O
```

Рис. 142

При входе в функцию предполагается, что справедливо одно из неблагоприятных условий (TEST = .TRUE.), так, что если оно действительно имеет место, то можно немедленно уйти на RETURN и вернуться в основную программу с правильным значением функции. В противном случае, когда происходит нормальный выход из цикла, переменной TEST присваивается значение .FALSE., поскольку не встретилось никакого неблагоприятного условия.

Параметры DØ-цикла передаются в функцию TEST через CØMMØN-блок. Напомним, что соответствие обеспечивается положением переменных в CØMMØN-блоке, а не их именами.

Если нам удалось успешно преодолеть 7 вызовов TEST, начинается печать, при которой после указания номера решения N печатаются Ø в пустых ячейках и «*» вместо каждого ферзя. Для этой цели используется вспомогательный массив K и две переменные «литерного» типа, K1 и K2.

ВОПРОСЫ:

1. Не лучше ли было бы поместить 7 «самых внутренних» DØ-циклов с вызовами TEST в еще один DØ-цикл, чем повторять каждый раз почти одинаковую инструкцию IF?
2. Почему не следует во всех DØ-циклах ссылаться на одну и ту же метку?

ОТВЕТЫ:

1. Нет, поскольку надо было бы, чтобы в Фортране допускалась рекурсия. Не вдаваясь в подробные объяснения, можно, чтобы убедиться в этом, уподобить структуру, в которой выполняется сначала 1 IF, затем 2, затем 3, ..., затем 7, структуре вычисления N! (что является классическим примером рекурсии), при котором вычисляют $1 \times 2 \times 3 \times \ldots \times N$.
2. Принципиально это возможно, но некоторые компиляторы (и среди них компилятор IBM) не допускают перехода с помощью GØ TØ на эту заключительную метку из иных циклов, чем самый внутренний. Диагностика, увы, не будет выдана, а программа зациклится.

8.3. РАЗНОЦВЕТНЫЕ КУБИКИ

(*условие задачи на стр.* 73)

Прежде всего эту задачу следует привести к «численному» виду, чтобы ее удобнее было решать на машине. Произвольно припишем каждому цвету свой номер: 1 — белому, 2 — зеленому, 3 — синему и 4 — оранжевому. Каждый кубик будет

представлен в машине столбцом из шести элементов в массиве KUBE (6, 4). Инструкция EQUIVALENCE (KUBE (1, 1), KUB1), дополненная инструкцией CØMMØN KUB1, KUB2, KUB3, KUB4, позволяет дать имя каждому отдельному кубику. Грани кубика перенумерованы, как показано на рис. 143, на котором мы видим, что грани, которые образуют видимые грани призмы, имеют номера 1, 2, 3 и 4.

Можно констатировать, что решение получено, если грани 1 у всех кубиков отличаются по цвету и если то же самое

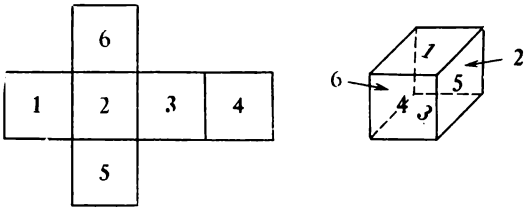


Рис. 143

условие выполняется для граней 2, 3, 4. Тогда для $1 \leqslant i \leqslant 4$ имеет место следующее соотношение:

$$2^{\text{цвет i-й грани кубика 1}} + 2^{\text{цвет i-й грани кубика 2}} + \\ + 2^{\text{цвет i-й грани кубика 3}} + 2^{\text{цвет i-й грани кубика 4}} = 30.$$

Действительно, так как значения цвета суть 1, 2, 3, 4, имеем

$$2^1 + 2^2 + 2^3 + 2^4 = 30$$

независимо от порядка показателей степени в том и только том случае, когда все цвета различны.

Чтобы найти все возможные решения, исходим из начального положения кубиков (введенного с перфокарты, что просто сделать) и последовательно поворачиваем каждый кубик. Краткие размышления и простые геометрические соображения позволяют учесть очевидную симметрию и избежать лишних поворотов. В частности, грани 5 и 6 не участвуют в решении. Поскольку у куба имеются три перпендикулярные оси, существуют три возможных вида вращения, что иллюстрируется рис. 144.

Повороты выполняются в подпрограмме RØTA. Ей дается имя кубика, который надо вращать, и направление вращения, обозначенное символически —1, 0 или 1, как показано на рис. 144. Эти значения выбираются так, чтобы при помощи арифметического IF можно было легко перейти к нужной части подпрограммы.

В каждом вращении есть два инварианта: грани, пересекаемые осью. Вращение производится перестановкой содержимого элементов, представляющих другие грани.

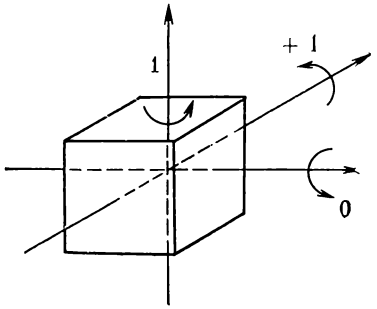


Рис. 144

Каждый из кубиков, за исключением последнего, вращается 4 раза в направлении 0, 3 раза в направлении —1 и 2 раза в направлении 1. Получаем таким образом 11 встроен-

```
      SUBROUTINE ROTA (KUB, DIRECT)
      DIMENSION KUB(6)
      INTEGER DIRECT
      IF (DIRECT) 1, 2, 3
2     L=KUB(1)
      KUB(1)=KUB(2)
      KUB(2)=KUB(3)
      KUB(3)=KUB(4)
      KUB(4)=L
      RETURN
1     L=KUB(1)
      KUB(1)=KUB(5)
      K=KUB(6)
      KUB(6)=L
      L=KUB(3)
      KUB(3)=K
      KUB(5)=L
      RETURN
3     K=KUB(6)
      KUB(6)=KUB(2)
      KUB(2)=KUB(5)
      KUB(5)=KUB(4)
      KUB(4)=K
      RETURN
      END
```

Рис. 145

ных DØ-циклов, в «теле» которых проверяется, не является ли решением то расположение, к которому мы только что пришли (рис. 146).

```
      DIMENSION KUBE(6,4)
      COMMON KUB1(6),KUB2(6),KUB3(6),KUB4(6)
      EQUIVALENCE (KUBE(1,1), KUB1(1))
      N = 0
      READ 1, KUB1, KUB2, KUB3, KUB4
      PRINT 444
      CALL PRINT
      DO 28 I1=1,2
      DO 27 I2=1,3
      DO 26 I3=1,2
      DO 25 I4=1,3
      DO 24 I5=1,4
      DO 23 I6=1,2
      DO 22 I7=1,3
      DO 21 I8=1,4
      DO 20 I9=1,2
      DO 19 I10=1,3
      DO 18 I11=1,4
      DO 17 I=1,4
      IF(2**KUB1(I)+2**KUB2(I)+2**KUB3(I)+2**KUB4(I).NE.30) GO TO 18
17    CONTINUE
      N=N+1
      PRINT 2, N
      CALL PRINT
18    CALL ROTA (KUB4,0)
19    CALL ROTA (KUB4,-1)
20    CALL ROTA (KUB4, 1)
21    CALL ROTA (KUB3,0)
22    CALL ROTA (KUB3,-1)
23    CALL ROTA (KUB3, 1)
24    CALL ROTA (KUB2,0)
25    CALL ROTA (KUB2,-1)
26    CALL ROTA (KUB2, 1)
27    CALL ROTA (KUB1,-1)
28    CALL ROTA (KUB1, 1)
      PRINT 222, N
      STOP 222
1     FORMAT (6I1)
2     FORMAT (///25H          РЕШЕНИЕ  НОМЕР  , I2, 1H:/)
222   FORMAT (///18H      НАЙДЕНО,      I2, 10H РЕШЕНИЯ   )
444   FORMAT (42H ИСХОДНОЕ   РАСПОЛОЖЕНИЕ      КУБИКОВ    : //)
      END
```

Рис. 146

```
      SUBROUTINE PRINT
      DIMENSION KUBE(6,4)
      EQUIVALENCE (KUBE(1,1), KUB1(1))
      COMMON KUB1(6),KUB2(6),KUB3(6),KUB4(6)
      DIMENSION COULER(2,4), COLOR(2,4,6)
      DATA COULER/4HBLAN,1HC,4HVERT,1H ,4HBLEU,1H ,4HORAN,2HGE/
      PRINT 10
      DO 1 K=1,6
      DO 1 J=1,4
      N=KUBE(K,J)
      DO 1 I=1,2
1     COLOR(I,J,K)=COULER(I,N)
      PRINT 12, ((COLOR(I,J,6),I=1,2),J=1,4)
      PRINT 11, (((COLOR(I,J,K),I=1,2),K=1,4),J=1,4)
      PRINT 12, ((COLOR(I,J,5),I=1,2),J=1,4)
      PRINT 10
10    FORMAT (1X,123(1H*)/)
11    FORMAT (/1X 4(4(A4,A3), ' * ')/)
12    FORMAT (/1X 4(7X A4,A3, 14X, ' * ') /)
      RETURN
      END
```

Рис. 147

```
ИСХОДНОЕ РАСПОЛОЖЕНИЕ КУБИКОВ:
*****************************************************************************
       VERT          *       ORANGE        *       BLANC         *       ORANGE        *
BLANC  BLEU  BLEU  ORANGE * BLANC  BLANC  ORANGE VERT  * BLANC  VERT  VERT  ORANGE * BLANC  VERT  BLEU  ORANGE *
       VERT          *       BLEU          *       BLEU          *       ORANGE        *
*****************************************************************************

          РЕШЕНИЕ  НОМЕР 1
*****************************************************************************
       VERT          *       BLANC         *       BLEU          *       ORANGE        *
BLEU   BLEU  BLANC ORANGE * ORANGE VERT  BLEU  BLANC  * BLANC  ORANGE VERT  VERT  * VERT  BLANC  ORANGE BLEU  *
       VERT          *       ORANGE        *       BLANC         *       ORANGE        *
*****************************************************************************

          РЕШЕНИЕ  НОМЕР 2
*****************************************************************************
       VERT          *       ORANGE        *       BLANC         *       ORANGE        *
ORANGE BLANC  BLEU  BLEU  * BLANC  BLEU  VERT  ORANGE * VERT  VERT  ORANGE BLANC * BLEU  ORANGE BLANC  VERT  *
       VERT          *       BLANC         *       BLEU          *       ORANGE        *
*****************************************************************************

       НАЙДЕНО  2 РЕШЕНИЯ
```

Рис. 148

Печать выделена в подпрограмму, чтобы можно было напечатать начальное расположение кубиков (рис. 147).

Чтобы исключить двойное индексирование (индекс с индексом) вида CØULER (KUBE (5, I)), используется вспомогательный массив CØLØR, в который предварительно пересылаются «номера цветов граней кубика». Разбиение по четыре литеры на слово годится для всех вычислительных машин, поэтому-то мы и предпочли его здесь.

Кроме того, как это видно на рис. 148, существует симметрия, поскольку одно из решений является «отражением» другого.

8.4. ЗАДАЧА О ШАХМАТНОМ КОНЕ [1]) (РЕШЕНИЕ В КОМПЛЕКСНЫХ ЧИСЛАХ)

(*условие задачи на стр.* 74)

В соответствии с указаниями в условии задачи определим шахматную доску как одномерный комплексный массив: CASE (64). Максимальное число полей продолжения CASFUI равно 8: 7 настоящих и еще одно, из которого делается ход. В дальнейшем нам станет ясна роль массива TABLE (64), который отождествляется с CASE в инструкции EQUIVALENCE. То же самое верно и для PRØV и CASEFUI. Программа воспроизведена на рис. 149.

Массив целых чисел ECHI (8, 8) служит лишь для печати каждого результата решения. Наконец, CHØIX (8), определенный с помощью DATA, представляет собой множество из 8 комплексных величин, которые можно использовать при переходе от одного поля к следующему.

A и B — координаты каждого исходного поля:

$$1 \leqslant A \leqslant 8, \quad 1 \leqslant B \leqslant 8.$$

Начнем с засылки нулей в массив, представляющий шахматную доску, и занесем в исходную позицию коня (CASE (1)) его координаты, определяемые A и B. Во время такой «подготовки почвы» воспользуемся случаем и запишем нули во вспомогательный массив ECHI. Настоящее начало программы — это цикл DØ 2 N = 2, 64.

Надо хорошо понять, что́ сейчас произойдет. Переменная CASE (n) содержит координаты поля, занятого конем, который сделал n — 1 ходов. Таким образом, ходы коня записываются от CASE (1) к CASE (64):

$$\text{CASE (n)} = \text{CASE (n + 1)} + \text{CHØIX (i)}, \quad 1 \leqslant i \leqslant 8.$$

[1]) Обсуждение этой задачи можно найти в статье Е. Гика («Наука и жизнь», № 5, 1976). — *Прим. ред.*

```
C       ХОД КОНЯ     ПОЛНОСТЬЮ КОМПЛЕКСНОЕ РЕШЕНИЕ
C
        INTEGER SOLUT, ECHI(8,8)
        LOGICAL BON
        COMPLEX CASE(64), CHOIX(8), CASFUI
        DOUBLE PRECISION TABLE(64), PROV
        EQUIVALENCE (CASE(1), TABLE(1)), (PROV, CASFUI)
        DATA CHOIX /(2.,1.),(1.,2.),(-1.,2.),(-2.,1.),(-2.,-1.),
     1                 (-1.,-2.),(1.,-2.),(2.,-1.)/
C
        SOLUT=0
        A=1
30      B=1
20      DO 1 I=2,64
1       CASE(I)=0
        CASE(1)=CMPLX(A,B)
        DO 9 I=1,8
        DO 9 J=1,8
9       ECHI(I,J)=0
        DO 2 N=2,64
        BON=.FALSE.
        MIN=8
        DO 3 I=1,8
        CASE(N)=CASE(N-1)+CHOIX(I)
        X=REAL(CASE(N))

        Y=AIMAG(CASE(N))
        IF(X.GT.8..OR.X.LT.1..OR.Y.GT.8..OR.Y.LT.1.) GO TO 3
        N1=N-1
        DO 4 K=1,N1
*ДИАГНОСТИКА* ПРОВЕРКА РАВЕНСТВА МЕЖДУ ЧИСЛАМИ НЕ INTEGER8   МОЖЕТ НЕ ИМЕТЬ СМЫСЛА.
        IF (TABLE(N).EQ.TABLE(K)) GO TO 3
4       CONTINUE
        BON=.TRUE.
        L=0
        DO 5 J=1,8
        CASFUI=CASE(N)+CHOIX(J)
        X=REAL(CASFUI)
        Y=AIMAG(CASFUI)
        IF (X.GT.8..OR.X.LT.1..OR.Y.GT.8..OR.Y.LT.1.) GO TO 5
        DO 6 K=1,N
*ДИАГНОСТИКА* ПРОВЕРКА РАВЕНСТВА МЕЖДУ ЧИСЛАМИ НЕ INTEGER8   МОЖЕТ НЕ ИМЕТЬ СМЫСЛА.
        IF (PROV.EQ.TABLE(K)) GO TO 5
6       CONTINUE
        L=L+1
5       CONTINUE
        IF (L.GE.MIN) GO TO 3
        MIN=L
        INDICE=J
3       CONTINUE
C
        IF (.NOT.BON) GO TO 1000
        CASE(N)=CASE(N-1)+CHOIX(INDICE)
2       CONTINUE
C
        SOLUT=SOLUT+1
C
        DO 7 K=1,64
        I=REAL(CASE(K))
        J=AIMAG(CASE(K))
7       ECHI(I,J)=K
        PRINT 8, SOLUT, A, B, ECHI
8       FORMAT (///' РЕШЕНИЕ  НОМЕР ',I3,' ВЫХОД ИЗ  X =',F4.0,
     1          ' И  Y =', F4.0// (8I4/))
10      B=B+1
        IF (B.LE.8.) GO TO 20
        A=A+1
        IF (A.LE.8.) GO TO 30
        STOP

1000    DO 17 K=1,N
        I=REAL(CASE(K))
        J=AIMAG(CASE(K))
17      ECHI(I,J)=K
        PRINT 1001, A, B, N, ECHI
1001    FORMAT (////' КАТАСТРОФА ПРИ ВЫХОДЕ ИЗ X =' F4.0,' И  Y =',
     1          F4.0, '   НА ПОЛЕ НОМЕР ', I3// (8I4/))
        GO TO 10
        END
```

Рис. 149

При каждом из n ходов вначале предполагается, что число полей продолжения MIN равно 8. На самом деле, мы видели, что их всегда меньше. MIN будет заменено в дальнейшем его подлинным значением.

В цикле DØ31 исследуются все возможные положения из n-го, в котором конь находится. При этом мы рискуем оказаться за пределами шахматной доски (если конь находится около края). В этом случае одна из составных частей комплексного числа CASE (N) будет меньше 1 или больше 8. Мы получаем X, вещественную часть (абсциссу), и Y, мнимую часть (ординату), из CASE (N), используя функции REAL и AIMAG. Если мы замечаем, что конь вышел за пределы шахматной доски, мы испытываем новое значение для CHØIX. Если ни одно не подходит, попадаем на проверку (IF.NØT.BØN), которая прямо отошлет к метке 1000 («КАТАСТРОФА»).

Теперь надо проверить, что на найденном поле (известно, что оно находится внутри доски) конь еще не был. Если бы он там был, мы бы нашли координаты этого поля у одного из n—2 пройденных полей. Поле с порядковым номером N обязательно отлично от поля с порядковым номером N — 1, потому что ни один из элементов массива CHØIX не равен нулю.

Кроме того, можно заметить, что конь попеременно переходит с белого поля на черное. Другими словами, номера полей одного цвета отличаются друг от друга на 2. Если N четно, достаточно просмотреть поля от 2 до N — 2 (независимо от четности N — 2). В противном случае надо просматривать от 1 до N — 2. Однако это справедливо лишь с поля номер 3, так как до этого мы бы сравнивали CASE (2) с ним самим. N2 является нижней границей DØ-цикла и должно иметь ту же четность, что и N — 2, т. е. ту же, что N.

Надо было бы писать

```
      |  DØ 4 K = N2, N1, 2
      |  IF (CASE (N).EQ.CASE (K)) GØTØ3
    4 |  CØNTINUE
```

К сожалению, некоторые компиляторы (и среди них тот, который мы использовали) проявляют здесь глупую ограниченность. Под ложным предлогом, что сравнивать два комплексных числа в том смысле, в котором мы сравниваем два вещественных, абсурдно, они отвергают даже операторы .NE. и .EQ.! Во всяком случае, мы обойдем эту трудность. Используя эквивалентность с массивом TABLE, мы все же произведем сравнение. Это возможно, так как вещественные и мни-

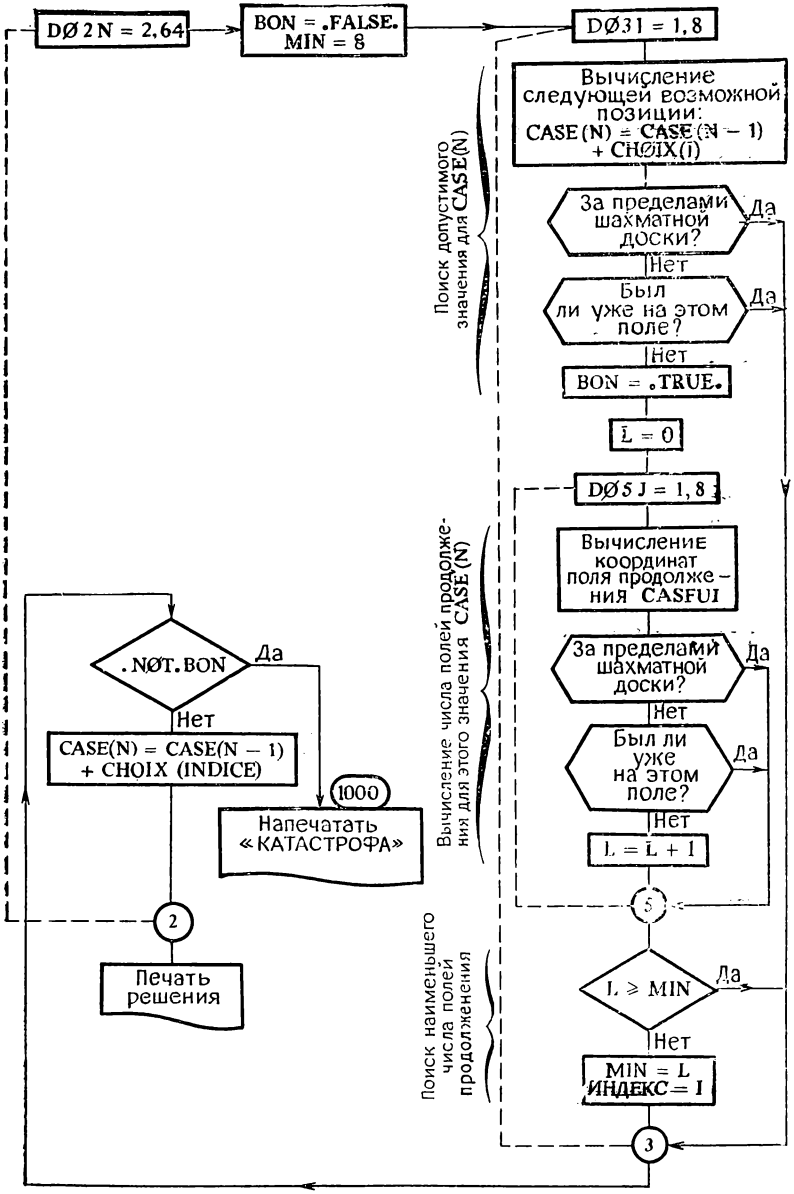

DØ 2 N = 2,64
BON = .FALSE.
MIN = 8
DØ 3 I = 1,8
Вычисление следующей возможной позиции:
CASE(N) = CASE(N – 1) + CHØIX(I)
Поиск допустимого значения для CASE(N)
За пределами шахматной доски?
Да
Нет
Был ли уже на этом поле?
Да
Нет
BON = .TRUE.
L = 0
DØ 5 J = 1,8
Вычисление числа полей продолжения для этого значения CASE(N)
Вычисление координат поля продолжения CASFUI
За пределами шахматной доски?
Да
Нет
Был ли уже на этом поле?
Да
Нет
L = L + 1
5
Поиск наименьшего числа полей продолжения
L ≥ MIN
Да
Нет
MIN = L
ИНДЕКС = I
3
.NØT. BON
Да
Нет
CASE(N) = CASE(N – 1) + CHOIX (INDICE)
1000
Напечатать «КАТАСТРОФА»
2
Печать решения

Рис. 150

мые части помещены в последовательные слова, подобно старшим и младшим разрядам переменной двойной длины.

Когда цикл DØ4 заканчивается нормально, получается одно правильное поле N. BØN принимает значение .TRUE.; это означает, что мы не зашли в тупик. Затем вычисляется число полей продолжения способом, аналогичным тому, ко-

```
КАТАСТРОФА ПРИ ВЫХОДЕ  ИЗ X =  3.  И Y =  5.   НА ПОЛЕ НОМЕР  61

 45  42  13  60   0  32  11  34
 14  55  46  43  12  35   0  31
 41  44  57  54  59   0  33  10
 56  15  52  47  36  25  30   0
 51  40   1  58  53  48   9  26
 16  19  50  37  24  27   6  29
 39   2  21  18  49   4  23   8
 20  17  38   3  22   7  28   5


РЕШЕНИЕ НОМЕР 21 ВЫХОД     ИЗ X =  3.  И Y =  6.
 40  31   6  57  52  59   8  63
  5  56  41  60   7  62  51  48
 30  39  32  53  58  49  64   9
 55   4  45  42  61  34  47  50
 38  29  54  33  46  43  10  19
  3  14   1  44  35  20  25  22
 28  37  16  13  26  23  18  11
 15   2  27  36  17  12  21  24
```

Рис. 151

торый использовался для нахождения CASE (N) и одновременной проверки его приемлемости. Блок-схема на рис. 150 пояснит организацию вычислений лучше всякого комментария. Надо запомнить индекс I, соответствующий наименьшему числу полей продолжения, так как придется снова вычислять нужное значение CASE(N).

Теперь надо переходить к определению координат следующего поля и т. д. вплоть до N = 64. Когда наступит этот момент, мы напечатаем найденное решение. Чтобы перейти от представления в комплексных числах к печати, воспользуемся способом, аналогичным тому, что мы применяли при решении задачи о числах по спирали, стр. 182.

Для каждой CASE (K) ($1 \leqslant K \leqslant 64$) координаты X и Y (действительная и мнимая части) служат для адресации к элементу ECHI (8, 8), в который будет помещено K. Здесь происходит что-то вроде преобразования времени в пространство. (CASE представляет собой последовательные положения коня в моменты t, t + 1, t + 2, ..., t + 63, в то время как ECHI представляет собой шахматную доску в своей физической реальности, с таким же расположением.)

Когда печать закончена, снова начинается вычисление для измененной отправной точки (изменяется B, потом A). В случае катастрофы печатается неполная доска и номер последнего достигнутого поля, как показано на рис. 151 (отправным было поле с координатами X = 3 и Y = 5, и катастрофа произошла на поле номер 61, но мы бы получили другие значения, если бы массив CHØIX был организован иначе, т. е. составляющие его величины были бы расположены в другом порядке), и затем проводится исследование для новой отправной точки.

Глава 10

НЕСКОЛЬКО СОВЕТОВ ПО ОПТИМИЗАЦИИ

Начинающий программист заботится прежде всего о том, чтобы программа работала, неважно как и сколько времени. В несколько приемов в этой книге мы пытались показать, как порой с помощью небольшого усилия и минимума здравого смысла, сохраняя, а иногда и улучшая точность результатов, можно уменьшить громоздкость программы в памяти, не увеличивая времени ее выполнения.

Несколько шире, чем в рамках наших упражнений, мы считаем возможным перечислить некоторые общие правила и дать некоторые советы, явившиеся плодом многолетнего опыта использования Фортрана на вычислительных машинах с разными характеристиками, который увеличивался не только при написании наших собственных программ, но главным образом при исправлении и отладке программ, написанных как профессиональными программистами, так и программистами-любителями.

Для большей ясности мы разделили наши советы на 5 рубрик, но надо признаться, что между этими рубриками нет четких граней и порой они пересекаются.

10.1. НАПИСАНИЕ ПРОГРАММ

Карты-продолжения

Используйте их лишь в крайних случаях (длинные инструкции FØRMAT или DATA) и, если это возможно, не заполняйте их без пробелов от 7 по 72 колонку. Очень часто возникает необходимость внести исправления, и вы будете очень рады, если вам не придется вставлять дополнительную карту. Если это возможно, нумеруйте карты-продолжения в порядке следования.

Имена переменных

Всегда избегайте возможной путаницы между 0 и Ø, I и l. Даже если вы правильно написали вашу программу, есть риск, что при перфорации могут появиться ошибки, не являющиеся синтаксическими, и поэтому компилятор не сможет

их обнаружить. Это приведет к тому, что результаты будут ошибочными, а причину ошибки установить будет трудно.

В ФОРТРАНЕ НЕТ СТАНДАРТНЫХ ИМЕН

Вы имеете право написать

```
GØ TØ5 = STØP4
DIMENSIØN READ(10)
DØ 2 I = READ(5)
```

(см. упражнение 7.7, стр. 71).

Избегайте использовать в индексах имена, не являющиеся неявно именами типа INTEGER. Тогда программу будет легче читать, а значит, и проще отлаживать, потому что тип переменной не будет вызывать сомнений.

Пробелы

Не бойтесь ими злоупотреблять. Гораздо легче читать программу, если каждая переменная, каждая инструкция Фортрана «разрежены» в своем контексте.

Карты-комментарии

Когда отладка программы заканчивается, вы способны быстро и без труда найти и исправить любой ее фрагмент. Несколько месяцев спустя положение изменится, и вы будете счастливы, если вам встретятся то там, то сям в программе хоть какие-то комментарии, характеризующие вычисления на данном участке программы.

Декларации

Ни один компилятор не требует, чтобы декларации обязательно предшествовали использованию описанных в них переменных. Однако проще сгруппировать их в начале программы.

Форматы

Неплохо было бы объединить их все вместе и поместить либо в самом начале программы, либо после деклараций, либо, если вам так угодно, в конце программы, расположив в порядке возрастания меток. Если потребуется иное или более аккуратное расположение информации при выводе, вам понадобится изменить форматы и тогда, чтобы их отыскать, не придется шарить по всей, возможно длинной, программе.

READ, PRINT, PUNCH

Вы имеете право написать READ (5, ...), чтобы ввести карты, WRITE (7, ...), чтобы их отперфорировать, и WRITE (6, ...), чтобы напечатать. Обычно это вполне приемлемо

только на вычислительной машине данного типа. На другой, напротив, вместо 5, 6, 7 надо писать 1, 2, 3, либо 60, 61, 62. Почему же не воспользоваться тем наследием Фортрана II, которое совместимо со всеми серьезными версиями Фортрана: READ, PUNCH, PRINT? К тому же эти инструкции выражены более явно. В нашем введении мы указали причины этого выбора, которого мы придерживались во всем тексте, и хотя для того, чтобы решить все наши задачи, нам пришлось использовать 5 очень разных вычислительных машин, никогда мы не сталкивались с трудностями на этом уровне.

Скобки

Чтобы рассеять заблуждение, отметим: НИЧТО НЕ СТОИТ ТАК ДЕШЕВО, КАК СКОБКИ. Можно даже сказать, что они помогают компилятору анализировать структуру инструкций. Во всяком случае, очевидно, что они полностью исчезают из результирующей программы. Вычислительное время, таким образом, не может ни в коей мере зависеть от числа скобок, имеющихся в символической программе.

Печать машинных команд

Задание соответствующего параметра в управляющей карте Фортрана позволяет запросить печать машинных команд, сгенерированных компилятором. Немногие программисты способны с пользой применять информацию такого рода. Следовательно, бессмысленно ее запрашивать, за исключением крайнего случая, когда требуется «дамп» — распечатка памяти при аварийном завершении программы.

Правила однородности

Лишь в очень старых версиях Фортрана II или в «микрокомпиляторах», используемых на «минимашинах», необходимо писать справа от знака равенства совершенно однородные арифметические выражения. На почти подавляющем большинстве современных машин компиляторы допускают инструкции вида:

```
|  | F = AINT (SIN (3.1415927/180 * (ID + MN/60.))),
```

которые гораздо проще писать и читать, чем инструкцию в следующем «каноническом» виде:

```
|  |  F = FLØAT (INT (SIN (3.1415927/180. * (FLØAT (ID)
|  |        + FLØAT (MN)/60.)))).
```

Однако будьте осторожны, этот путь может вас завести слишком далеко!

```
SQRT (J ** 2 + I/K) неверно, надо писать
SQRT (FLØAT (J ** 2 + I/K)),
```

потому что аргументы функций и подпрограмм не подвергаются неявным преобразованиям.

Кроме того, в константах с двойной точностью должен обязательно присутствовать показатель степени, которому предшествует буква D.

10.2. УПРАВЛЕНИЕ ПАМЯТЬЮ

Переменные с индексами

Вообще на вычисление индекса затрачивается довольно много времени при выполнении программы. Если необходимо использовать какой-нибудь отдельный элемент массива в нескольких подряд стоящих формулах или проверках, полезно для экономии времени заранее заслать его значение во временную переменную того же типа, но без индекса. Например, вместо того, чтобы писать

```
      A(I, J) = C(I1, J, K) + B
      . . . . . .
      IF (C(I1, J, K).EQ. . . .
```

лучше написать

```
      X = C(I1, J, K)
      A(I, J) = X + B
      . . . . . .
      IF (X.EQ. . . .
```

Если нам необходимо обрабатывать поэлементно массив с несколькими индексами, установление эквивалентности с одноиндексным массивом приведет к заметной экономии времени (см., например, упражнение 3.7 [1]) на стр. 106).

Аналогичный случай нам встретится дальше в связи с вводом-выводом массивов.

Размер памяти

В подпрограммах или функциях для массивов, имена которых встречаются в списке формальных параметров, память не отводится и, следовательно, они не дублируются в вызывающей и вызываемой подпрограммах, потому что передается лишь их начальный адрес. Поэтому нет никаких ограничений на место в памяти при передаче массивов в подпрограммы.

---

[1]) В примечании к предыдущему упражнению дана количественная оценка этой рекомендации. — *Прим. ред.*

10.3. ВВОД-ВЫВОД

Полные массивы

Лучше писать

```
DIMENSIØN A(10, 30)
WRITE(1, 10) A
```

чем

```
DIMENSION A(10, 30)
WRITE(1, 10)((A(I, J), I = 1, 10), J = 1, 30).
```

Подпрограммы на Фортране быстрее обработают неявную последовательность индексов, чем подробно вами расписанную. Тем не менее надо обратить внимание на порядок следования индексов. (Индекс, написанный первым, изменяется в первую очередь.)

С форматом или без?

Оператор FØRMAT играет двоякую роль: он обеспечивает размещение (верстку) на носителе (бумаге, ленте) и определяет преобразования между внутренним и внешним представлениями информации. Строго говоря, формат необходим лишь в случае, когда эту информацию должен читать человек либо когда есть намерение использовать созданные файлы на вычислительных машинах другого типа. Если же речь идет только о хранении промежуточных результатов для последующих вычислений на той же машине, отсутствие формата позволит выиграть время, место на внешнем носителе и точность численных значений.

Так, на ЭВМ IBM 360/370 4 байтов достаточно для хранения переменной вещественного типа, в то время как при использовании спецификации E 15.7 понадобилось бы 15 байтов.

Разумеется, мы получаем двойной выигрыш (во времени и в точности), потому что бесформатная запись эффективнее как при вводе, так и при выводе информации. Единственное неудобство состоит в том, что обычно при повторном чтении список ввода-вывода должен иметь ту же длину, что и при создании некоторой записи.

Литеры перехода со строки на строку

Сколько чистых страниц (за исключением нескольких цифр вверху слева) было растрачено зря только потому, что программист забыл о литере перехода со строки на строку, а значение результата имело на один разряд больше, чем он предусмотрел! Ведь совсем нетрудно начинать все инструкции FØRMAT с «1X» (или «1 H0», «1 H1», ...).

Надо ценить бумагу

Если при отладке можно еще простить вывод значений по одному на строку, то во всех других случаях достойна осуждения программа, в которой не используется вся доступная ширина бумаги. Минимум обычно составляет 120 позиций, более часто 132, существует вариант со 136 литерами в строке [1]).

В большинстве существующих систем программист может с помощью соответствующего параметра в управляющей карте задать максимальное число страниц или строк, которые он хочет напечатать. Отлаживаемая программа может зациклиться, что может вызвать бесконечное повторение одной строки или группы строк, лишенных интереса и значения. Разумное использование этого параметра ограничит возможные убытки.

Печать обходится дорого

А здесь мы выдвинем обвинения против литеры «+», запрещающей переход со строки на строку. Известно, что эта литера позволяет печатать строку в несколько приемов, так как блокируется переход на новую строку. Но на уровне выполнения программы это потребует столько же времени центрального процессора, сколько занимает печать нескольких строк. Скорость печати при этом резко снизится.

А нужно ли много печатать?

В вычислительных центрах можно встретить немало пользователей, которые наваливают груды бумаги, полученной при систематической распечатке огромных файлов или всех промежуточных результатов при долгих вычислениях. Безусловно, это надежный способ защиты информации на случай, если обвалится потолок, но, что касается научной ценности метода, то она почти равна нулю. В самом деле, немыслимо проанализировать такую массу цифр. Если мы хотим убедиться в отсутствии непредусмотренных изменений в файле, более рациональна специальная обработка — частичная печать, т. е. печать только тех мест, где есть отклонения. Конечно, надо уметь выявить эти сомнительные места. Но если мы способны это сделать визуально, то почему же нельзя формализовать соответствующие критерии и запрограммировать их.

Кроме того, существуют графопостроители (BENSON, CALCOMP, ...), которые в экономной, сжатой и куда более наглядной форме могут представить информацию, содержа-

[1]) В отечественных АЦПУ допускается обычно 128 литер в строке. — *Прим. ред.*

щуюся во многих страницах результатов. Наконец, системы «микрофишей» позволяют экономное и компактное хранение информации.

10.4. ОБЩИЕ АСПЕКТЫ ОПТИМИЗАЦИИ

Не следует одну и ту же программу транслировать ДВАДЦАТЬ РАЗ

В Фортране предполагается, что модульная организация позволяет по отдельности отладить некоторые куски программы в форме подпрограмм. Но если они правильные, то зачем их заново компилировать при каждом проходе, проще хранить их в виде «объектного модуля». Мы бы получили выигрыш во времени компиляции, в бумаге, и, что самое главное, возросла бы надежность результатов, так как не было бы риска потерять или перемешать карты (в двоичном виде несчастные случаи подобного рода обнаруживаются в момент загрузки).

Не злоупотребляйте подпрограммами

Время обращения к подпрограмме отнюдь не является пренебрежимым. Короткие, часто повторяющиеся фрагменты (помещенные во вложенные DØ-циклы, например) лучше непосредственно писать, даже если это приходится делать несколько раз в разных частях программы. Возможно, получившаяся при этом программа станет несколько более громоздкой, но время ее выполнения будет короче.

Промежуточные результаты

Неверно утверждать, что выгоднее заново вычислить промежуточный результат, чем поместить его раз и навсегда в рабочую область памяти. Время доступа к памяти почти всегда меньше времени выполнения арифметической операции, а поскольку операция требует по меньшей мере двух операндов и, следовательно, двух обращений к памяти, то совершенно непонятно, на какую экономию можно надеяться. Таким образом, лучше написать

```
      Z = 3.1415927/180
      DØ 1 M = 1,51,10
      F = SIN(Z * ...,
```

чем

```
      DØ 1 M = 1,51,10
      F = SIN(3.1415927/180. * ...).
```

Кроме того, в этом частном случае имеется другая причина вычислять Z, о которой мы узнаем позже.

Чрезмерная оптимизация

Выше мы советовали, насколько это возможно, избегать обращения к переменным с индексами. Слепое следование этому совету могло бы привести, например, к тому, что, оперируя с матрицами, мы отказались бы воспользоваться DØ-циклом. Такой случай известен, и мы знаем программу из приблизительно 1200 карт, оперирующую с матрицами, каждый элемент которых имел имя: A11, A12, A13, ..., A21, A22, ... и каждая инструкция переписывалась столько раз, сколько потребовалось, при этом каждый раз происходили необходимые изменения имен переменных. Потребовалось более 2 минут компиляции и всего менее 2 секунд вычислений. Переписанная более рациональным способом, эта программа компилировалась менее чем за 5 секунд, а время ее выполнения едва лишь удвоилось!

Будьте проще!

Некий английский программист написал

```
|  | A = SQRT (B ** 2),
```

чтобы получить абсолютное значение B, не подозревая о существовании стандартной функции ABS.

Не усложняйте!

Если вы должны найти синус угла, большего 2π радиан, напрасно сводить его предварительно к углу в интервале от 0 до 2π, библиотечная подпрограмма Фортрана сама сделает это за вас, если только вы остаетесь в пределах разумного числа оборотов.

Ну, а если вам обязательно хочется сделать это преобразование (что сводится к нахождению остатка от деления), то существует для этого функция MØD (для целых чисел) или AMØD (для вещественных), которая избавит вас от проделывания длинного ряда вычитаний.

Логический IF или арифметический IF?

В душе мы предпочитаем логический IF, поскольку его запись более наглядна, чем у его арифметического собрата. К тому же он порой дает выигрыш во времени вычислений. Так,

```
|  | IF (I.LT.0) GØTØ 5
|  | IF (I.EQ.0) GØTØ 6
```

предпочтительнее

```
|   | IF (I) 5, 6, 7
```

особенно если I часто бывает отрицательным. В случае если бы было

```
IF (I.LT.6) GØTØ 5
IF (I.EQ.27) GØTØ 5,
```

компактная запись

```
IF (I.LT.6.ØR.I.EQ.27) GØTØ 5
```

была бы чаще всего менее удачной, потому что обычно прежде всего выполняются все вычисления, а затем проверки, что приводит к потере времени.

Когда имеется каскад проверок, важно записать их в убывающем порядке вероятности. Например, предположим, что требуется преобразовать программу на Фортране, отперфорированную в коде BCD (на IBM 026), в код EBCDIC (типичный для IBM 029)[1]). Надо преобразовать литеры (,), +, /, =. Запишем

```
|   |DATA BLANC, PARG, PARD, PLUS, APØS, EGAL/1H /,
1|   |  1H (, ...
|   |DØ 1 I = 1,80
|   |IF (CARTE (I) .EQ.BLANC) GØTØ 1
|   |IF (CARTE (I) .EQ.PARG) CARTE (I) = ...
|   |...
1|   |CØNTINUE
```

По опыту известно, что карты с программой на Фортране имеют много пробелов. Очевидно, программа, в которой в первую очередь обрабатываются пробелы, затем другие литеры в порядке убывания частоты, будет более эффективной.

Правильное использование DØ-циклов

Некоторые компиляторы располагают оптимизаторами, которые выносят за пределы DØ-циклов вычисление выражений, или подвыражений, не зависящих от параметра рассматриваемого цикла. Поскольку во многих случаях неизвестно в деталях, что эти оптимизаторы делают, предпочтительнее

[1]) IBM 026 и IBM 029 — устройства для подготовки перфокарт. — *Прим. ред.*

проделывать операции подобного рода самим. Так, в следующем примере (намеренно искусственном)

```
      DØ 1 I = 1, 10
      DØ 1 J = 1, 20
      C (I) = A (I) * B
      C (J) = C (J) + A (I)
        . . . . . .
1     CØNTINUE
```

Напрасно повторять вычисление C(I), поскольку I не изменяется, и лучше было бы написать

```
      DØ 1 I = 1, 10
      C (I) = A (I) * B
      DØ 1 J = 1, 20
      C (J) = C (J) + A (I)
        . . . .
1     CØNTINUE
```

10.5. ВЫЧИСЛИТЕЛЬНЫЕ ОПЕРАЦИИ

Сложение и вычитание

Рассмотрим выражение A = B + C, где B = 12345,6 и C = 0,0075. Выполним вычисления на вычислительной машине IBM 360 или 370. Получим A = 12345,6 и, не вникая в существо дела, осудим компилятор IBM за то, что он не выполняет инструкций, которые мы потрудились написать. Тем более что, если мы запускаем эту же программу на вычислительной машине CONTROL DATA 6400, находим A = = 12345,6075.

На деле это происходит из-за того, что в вычислительной машине IBM вещественные числа представляются 32 битами, что соответствует 6—7 десятичным знакам мантиссы, в то время как в CONTROL DATA используются слова из 60 битов, что дает 14 десятичных разрядов. На IBM сложению предшествует выравнивание порядков:

```
1  2  3  4  5  6        |
                0  0  | 7  5
-------------------------------
1  2  3  4  5  6  0  | 7  5
```

и мы видим, что результат получился правильный, но он зависит от используемой вычислительной машины.

Умножение и деление

«...например, при вычислении a/(b·c) можно проявить леность и, не вникая в подробности, написать A/(B * C) вместо A/B/C...»

Хотя это утверждение может показаться соблазнительным, мы не посмели бы одобрить его безоговорочно. И прежде всего потому, что на всех вычислительных машинах деление длиннее умножения, затем потому, что записанное в бесскобочном виде арифметическое выражение труднее для восприятия при чтении программы.

Возведение в целую степень

Хотя A * *4. и A * * 4 на первый взгляд дают одинаковые результаты, в Фортране эти два выражения подвергнутся различной обработке. Первое будет вычислено при помощи логарифма, так как 4 — вещественное число, в то время как второе будет вычислено последовательными умножениями, так как 4 — целого типа. И времени на это уйдет меньше. С другой стороны, если A отрицательно или равно нулю, в первом случае вычисления вообще нельзя будет выполнить (логарифм не вычислим).

Особый случай, встречающийся время от времени при манипуляциях с матрицами, требует вычисления $(-1)^n$ (где n— индекс переменной). Если это переписать, не подумав, получим

```
|  | K = (-1) ** N.
```

Когда значение N велико, мы делаем большое число умножений, в то время как, учитывая что в результате мы всегда получим либо —1, либо +1, эффективнее было бы написать

```
|  | K = 1
|  | IF (MØD (N, 2).EQ.1) K = - 1.
```

ОГЛАВЛЕНИЕ

М. Дрейфус, К. Ганглоф
Практика программирования на Фортране

Редакторы А. А. Бряндинская и И. А. Маховая
Художник А. В. Шипов
Художественный редактор В. И. Шаповалов
Технический редактор Н. И. Борисова
Корректоры Л. В. Байкова, Л. Д. Панова

ИБ № 1015

Сдано в набор 07.03.78. Подписано к печати 20.09.78. Формат 60×90¹/₁₆. Бумага типографская № 1. Гарнитура латинская. Печать высокая. Объем 14 п. л., бум. л. 7. Уч.-изд. л. 11,49. Изд. № 1/9485. Тираж 79 000 экз. Заказ № 1054. Цена 90 коп.
Издательство «Мир»
Москва, 1-й Рижский пер., 2.

Ордена Трудового Красного Знамени
Ленинградская типография № 2 имени Евгении Соколовой
«Союзполиграфпрома» при Государственном комитете СССР
по делам издательств, полиграфии и книжной торговли,
198052, Ленинград, Л-52, Измайловский проспект, 29.