В. Е. АЛЕКСЕЕВ А. С. ВАУЛИН Г. Б. ПЕТРОВА

Вычислительная техника в инженерных и экономических расчетах

сборник задач и упражнений

для вузов

издательство Высшая Школа

В. Е. АЛЕКСЕЕВ, А. С. ВАУЛИН, Г. Б. ПЕТРОВА

Вычислительная техника в инженерных и экономических расчетах

Сборник задач и упражнений

Под редакцией д-ра техн. наук,
проф. А. В. Петрова

Допущено Министерством высшего
и среднего специального образования СССР
в качестве учебного пособия для студентов
высших учебных заведений

МОСКВА «ВЫСШАЯ ШКОЛА» 1984

ББК 65.9(2)21
А47
УДК 681.332.6

Рецензенты:
кафедра «Вычислительная техника» Московского института электронного машиностроения (зав. кафедрой — *П. П. Сыпчук*), д-р техн. наук, проф. *В. И. Дракин*.

Алексеев В. Е., Ваулин А. С., Петрова Г. Б.

А47 Вычислительная техника в инженерных и экономических расчетах. Сборник задач и упражнений: Учеб. пособие для вузов./Под ред. *А. В. Петрова*. — М.: Высш. шк., 1984.— 136 с., ил.

35 к.

В пособии рассматриваются типовые задачи по программированию на алгоритмических языках ФОРТРАН-IV и ПЛ/I, приводятся задания для самостоятельной разработки схем алгоритмов и программ, выполняемых на ЭВМ третьего поколения. Для большинства задач даются ответы.

Приведенные методические указания по разработке программ тесно связаны с материалом учебника «Вычислительная техника в инженерных и экономических расчетах» (см. № 132).

А $\frac{2405000000-329}{001(01)-84}$ 129—84

ББК65.9(2)21
6Ф7

ПРЕДИСЛОВИЕ

Рост научно-технического прогресса, усиление роли науки — объективная тенденция современного этапа коммунистического строительства. Экстенсификация экономического развития сменяется интенсификацией, осуществляемой при широком использовании достижений науки. Характерная черта научно-технической революции — стремительно возрастающая роль ЭВМ во всех областях человеческой деятельности. Без ЭВМ в настоящее время немыслимо решение широкого круга инженерных, научных, экономических и управленческих задач. Использование ЭВМ в многовариантных и оптимизационных расчетах, управление технологическими процессами и организационно-экономическом управлении народным хозяйством, автоматизации проектирования сложных объектов позволяет получить значительный экономический эффект за счет сокращения сроков проектирования, повышения качества проектных работ, что, в свою очередь, приводит к снижению себестоимости и повышению производительности труда.

Следовательно, темпы научно-технического прогресса в ведущих областях народного хозяйства в огромной степени будут определяться качеством и номенклатурой вычислительных средств и их программным обеспечением. Такое широкое внедрение высокопроизводительных средств вычислительной техники в различные сферы народного хозяйства диктует необходимость подготовки специалистов, сочетающих знания своей специальности с математическими методами решения задач и навыками использования для этих целей ЭВМ.

Сборник задач имеет практическую направленность и дополняет учебник «Вычислительная техника в инженерных и экономических расчетах» под ред. А. В. Петрова. Материал сборника и учебника соответствует программе одноименного курса и методически взаимоувязан.

В сборнике задач с единых позиций рассмотрены способы алгоритмизации задач, правила применения языковых средств ФОРТРАНа и ПЛ/1 и приемы программирования, что является его отличительной особенностью. Он содержит необходимые сведения теоретического характера, иллюстрирован конкретными примерами, что вполне достаточно для разработки алгоритмов и программ реальных задач. Единая методическая основа данного пособия и доступность изложения материала позволяет студентам самостоятельно изучать вопросы алгоритмизации и программирования инженерных задач.

Пособие состоит из трех разделов. Первый раздел включает задачи, связанные с разработкой структур алгоритмов и наиболее часто используемые приемы программирования. В следующих двух разделах излагаются синтаксические конструкции языков ФОРТРАН и ПЛ/1, правила организации структур программ, различных конфигураций. Нумерация задач дана по разделам.

Задачи, отмеченные символом ▲, имеют ответ, а символом ● — даны с решением.

Авторы выражают благодарность коллективу кафедры «Вычислительная техника» Московского института электронного машиностроения и д-ру техн. наук, проф. В. И. Дракину, взявшим на себя труд по рецензированию настоящего пособия и сделавшим ряд ценных замечаний.

Авторы ставили своей целью помочь читателям в практической работе при изучении вопросов алгоритмизации и программирования на алгоритмических языках ФОРТРАН-IV и ПЛ/1. Насколько это удалось, авторы надеются судить по отзывам и замечаниям, которые можно направлять по адресу: *101430, Москва, ГСП-4, Неглинная ул., д. 29/14, изд-во «Высшая школа», за что авторы заранее благодарны.*

Авторы

ВВЕДЕНИЕ

Процедура подготовки задач к решению на цифровых вычислительных машинах до настоящего времени остается достаточно сложной и трудоемкой и включает в себя следующие основные этапы:

1. Постановка задачи (задача, которую предстоит решать на ЭВМ, формулируется пользователем или получается им в виде задания).
2. Разработка или выбор существующего алгоритма решения задачи.
3. Выбор численных методов решения для отдельных участков алгоритма с учетом заданной точности, быстродействия и т. д.
4. Составление схемы программы, представляющей собой графическое изображение программы в виде отдельных блоков и их связей (схема в значительной степени облегчает знакомство с программой и упрощает ее запись на алгоритмическом языке).
5. Написание программы (по схеме программы, используя алгоритмический язык, составляют программу и записывают ее на бланках).
6. Подготовка исходных данных (в большинстве случаев для программы требуются исходные данные, которые также записываются на бланках).
7. Перенос программы и исходных данных на носитель информации (перфокарты, перфоленты).
8. Формирование задания, которое описывается на языке управления заданиями с помощью управляющих операторов.
9. Отладка программы (отладка программы может подразделяться на ряд подэтапов, включая визуальный контроль, синтаксический контроль и т. д.).

Особое место при подготовке задач к решению на ЭВМ занимают разработка или выбор алгоритма, написание программы на алгоритмическом языке и ее отладка. Описание алгоритма представляет собой общую схему решения задач. Различают способы описания алгоритмов словесный, операторный и схемный. Наибольшее распространение в настоящее время получил схемный способ, при котором вычислительный процесс расчленяется на отдельные операции, отображающиеся в виде условных графических блочных символов. С 1.07.81 введены ГОСТ 19002—80 и 19003—80, определяющие правила выполнения схем и перечень блоков, их наименования, форму и размеры.

Эффективное использование ЭВМ при решении научно-технических задач требует умения программировать на алгоритмических языках. Наиболее распространены в настоящее время проблемно-ориентированные алгоритмические языки ФОРТРАН, ПЛ/1 и КОБОЛ. Алгоритмический язык ФОРТРАН характеризуется простотой использования, близостью записи конструкций к обычной математической записи, большими возможностями ввода — вывода информации, получением программ после трансляции, близких к оптимальным, простотой и наглядностью определения ошибок по сообщениям транслятора, наличием большой библиотеки научных подпрограмм, включающей практически все численные методы решения. Алгоритмический язык ПЛ/1, обладая всеми преимуществами языков ФОРТРАН, АЛГОЛ-60, КОБОЛ, имеет много других, присущих именно ему достоинств. Это позволяет расширить область его применения (например, использовать для решения задач экономических, планово-статистических, моделирования, логических, в реальном масштабе времени и для разработки программного обеспечения ЭВМ). Язык ПЛ/1 допускает блочную структуру программ, хорошо приспособлен для обработки символьных текстов и организации информации в списки.

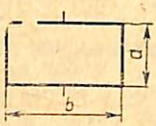
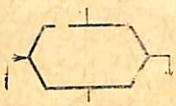
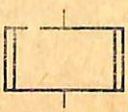
1. РАЗРАБОТКА АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

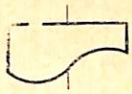
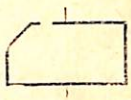
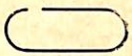
Подготовка задачи для решения на ЭВМ состоит из следующих этапов: математической формулировки условия задачи, выбора численного метода ее решения, разработки схемы алгоритма, составления программы на алгоритмическом языке и ее отладки. В настоящем сборнике условия большинства задач представлены в математической формулировке с указанием численного метода решения. Поэтому отпадает необходимость в выполнении первых двух этапов и можно непосредственно приступать к разработке схем алгоритмов решения задачи на ЭВМ.

Алгоритм — некоторая конечная последовательность предписаний (правил), однозначно определяющих процесс преобразования исходных и промежуточных данных в результат решения задачи. Таким образом, при разработке алгоритма решения задачи математическая формулировка является основой для определения последовательности действий, приводящих к получению искомого результата. Алгоритм должен отвечать требованиям детерминированности результатов, массовости применения для исходных данных, результативности, обеспечивающей получение результата через конечное число шагов.

Наиболее наглядный способ представления алгоритмов — их изображение в виде схем — последовательности блоков, предписывающих выполнение определенных функций, и связей между ними. Внутри блоков указывается поясняющая информация, характеризующая выполняемые ими действия. Блоки схемы обычно имеют сквозную нумерацию. Конфигурацию и размер блоков, а также порядок построения схем определяют ГОСТ 19002—80 и 19003—80. В табл. 1.1 приведены некоторые наиболее часто употребляемые блоки и даны пояснения к ним.

Таблица 1.1

Название символа	Символ	Примечание
Процесс		Вычислительное действие или последовательность вычислительных действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисление по подпрограмме или стандартной подпрограмме

Название символа	Символ	Примечание
Документ		Вывод данных, печать результатов
Перфокарта		Ввод данных с перфокарт или вывод данных на перфокарты
Соединитель		Разрыв линий потока
Пуск, останов		Начало, конец, останов, вход и выход в подпрограммах
Комментарий		Пояснения, содержание подпрограмм, формулы

По характеру связей между блоками различают алгоритмы линейной, разветвляющейся и циклической структуры.

1.1. АЛГОРИТМЫ ЛИНЕЙНОЙ СТРУКТУРЫ

Алгоритм линейной структуры (линейный алгоритм) — алгоритм, в котором блоки выполняются последовательно друг за другом. Такой порядок выполнения блоков называется *естественным*.

● **1.1.** Вычислить высоты треугольника со сторонами a, b, c по формулам

$$h_a=(2/a)\sqrt{p(p-a)(p-b)(p-c)};$$

$$h_b=(2/b)\sqrt{p(p-a)(p-b)(p-c)};\quad h_c=(2/c)\sqrt{p(p-a)(p-b)(p-c)},$$

где $p=(a+b+c)/2$.

Для решения любой нетривиальной задачи может иметь место несколько алгоритмов, приводящих к получению результата ее решения. Из всех возможных алгоритмов следует выбирать наилучший в смысле некоторого критерия. Часто в качестве критерия используют либо оценку точности решения задачи, либо затраты времени на ее решение, либо некоторый интегральный критерий, включающий оценки и точности, и затрат времени.

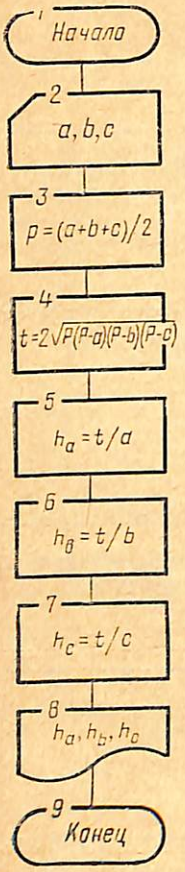


Рис. 1.1

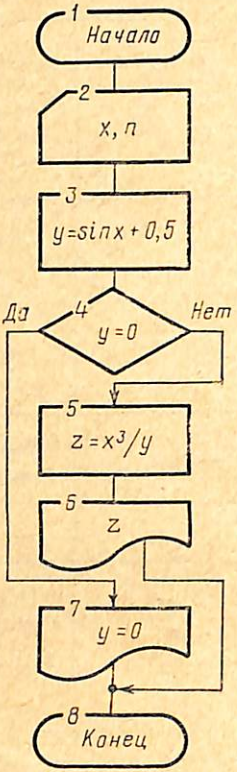


Рис. 1.2

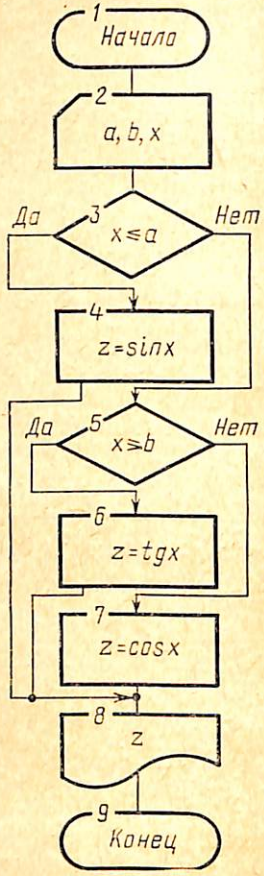


Рис. 1.3

При организации решения задачи для исключения повторений вычислять высоты следует не по приведенным выше формулам непосредственно, а используя промежуточную переменную $t=2\sqrt{p(p-a)(p-b)(p-c)}$, тогда $h_a=t/a$, $h_b=t/b$, $h_c=t/c$.

С учетом сказанного схема алгоритма решения задачи будет иметь вид, представленный на рис. 1.1.

1.2. Вычислить площадь поверхности $s=\pi(R+r)l+\pi R^2+\pi r^2$ и объем $v=(1/3)\pi(R^2+r^2+Rr)h$ усеченного конуса.

1.3. Вычислить координаты центра тяжести трех материальных точек с массами m_1, m_2, m_3 и координатами (x_1, y_1), (x_2, y_2), (x_3, y_3)

по формулам $x_c=(m_1x_1+m_2x_2+m_3x_3)/(m_1+m_2+m_3)$, $y_c=(m_1y_1+m_2y_2+m_3y_3)/(m_1+m_2+m_3)$.

1.4. Вычислить координаты точки, делящей отрезок a_1a_2 в отношении $n_1:n_2$, по формулам $x=(x_1+\gamma x_2)/(1+\gamma)$, $y=(y_1+\gamma y_2)/(1+\gamma)$, где $\gamma=n_1/n_2$.

1.5. Вычислить медианы треугольника со сторонами a, b, c по формулам

$$m_a=0{,}5\sqrt{2b^2+2c^2-a^2},\quad m_b=0{,}5\sqrt{2a^2+2c^2-b^2},$$

$$m_c=0{,}5\sqrt{2a^2+2b^2-c^2}.$$

1.6. Вычислить значение функции $y=ae^{-ax}\sin\omega x$ при $x=(\pi/2-\varphi)/\omega$.

1.7. Вычислить значения функций

$$y=(e^{-x_1}+e^{-x_2})/2 \text{ и } z=(a\sqrt{x_1}-b\sqrt{x_2})/c,$$

где $x_1=(b+\sqrt{|b^2-4ac|})/(2a)$, $x_2=(b-\sqrt{|b_2-4ac|})/(2a)$.

1.8. Определить высоту треугольника, если его площадь равна s, а основание больше высоты на величину a.

1.2. АЛГОРИТМЫ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

На практике редко удается представить схему алгоритма решения задачи в виде линейной структуры. Часто в зависимости от каких-либо значений промежуточных результатов необходимо организовать вычисление либо по одним, либо по другим формулам, т. е. в зависимости от выполнения некоторого логического условия вычислительный процесс должен идти по одной или другой ветви. Алгоритм такого вычислительного процесса называется *алгоритмом разветвляющейся структуры*. В общем случае количество ветвей в таком алгоритме разветвляющейся структуры не обязательно равно двум.

● **1.9.** Вычислить значение функции $z=x^3/y$, где $y=\sin nx+0{,}5$.

Казалось бы, решение этой задачи можно описать алгоритмом линейной структуры. Однако для удовлетворения свойств массовости и результативности алгоритма необходимо, чтобы при любых исходных данных был получен результат или сообщение о том, что задача не может быть решена при заданных исходных данных. Действительно, если $y=0$, задача не может быть решена, так как деление на 0 невозможно. Поэтому в алгоритме необходимо предусмотреть этот случай и выдать в качестве результата информацию о том, что $y=0$. Таким образом, рассматриваемый вычислительный процесс должен иметь две ветви: в одной, если $y\neq0$, необходимо вычислить и отпечатать значение переменной z, а в другой — вывести на печать информацию, что $y=0$.

Этот вычислительный процесс можно описать условным выражением:

$$\begin{cases}\text{вычислить } z=x^3/y, \text{ если } y\neq0;\\ \text{вывести } Y=0, \text{ если } y=0.\end{cases}$$

Схема алгоритма решения этой задачи представлена на рис. 1.2, где после условного блока *4* располагаются блоки сначала одной

ветви (блоки *5*, *6*), а затем второй ветви (блок *7*). Поскольку после выполнения блоков первой ветви нет надобности выполнять блоки второй ветви, осуществляется переход сразу к концу алгоритма (к блоку *8*).

В алгоритме дважды нарушается естественный порядок выполнения блоков: 1) при проверке условия $y=0$ (условный переход); 2) после выполнения блоков первой ветви (безусловный переход).

● **1.10.** Вычислить значение функции

$$z=\begin{cases}\sin x, & \text{если } x \leqslant a;\\ \cos x, & \text{если } a < x < b;\\ \operatorname{tg} x, & \text{если } x \geqslant b.\end{cases}$$

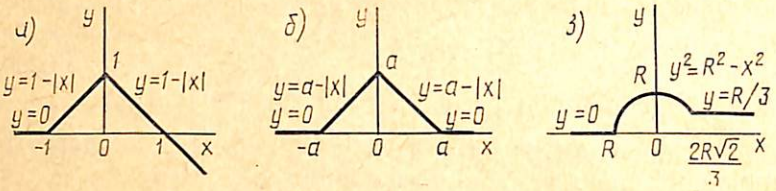


Рис. 1.4

Здесь вычислительный процесс имеет три ветви. С помощью условного блока можно проверить выполнение только условия, по которому будет определен выбор выражения для реализации одной ветви. Поэтому, чтобы установить, по какой из двух оставшихся ветвей должен идти вычислительный процесс в случае невыполнения первого условия, необходимо использовать еще один условный блок. Схема этого алгоритма представлена на рис. 1.3. Блок *3* проверяет условие $x \leqslant a$ и в случае его выполнения осуществляет переход к блоку *4*, вычисляющему $z=\sin x$. Если $x>a$, то блок *5* проверяет условие $x \geqslant b$. Если это условие выполняется, то осуществляется переход к блоку *6*, вычисляющему $z=\operatorname{tg} x$. В противном случае x лежит в интервале между a и b и происходит переход к блоку *7*, вычисляющему $z=\cos x$. После вычислений по любой из формул осуществляется переход в общую ветвь к блоку печати.

1.11. Вычислить корни квадратного уравнения $ax^2+bx+c=0$. Если $d=b^2-4ac \geqslant 0$, то корни действительные; следовательно, необходимо вычислить $x_{1,2}=e \pm f$. Если $d<0$, то корни мнимые; следовательно, необходимо вычислить e и f по формулам $e=-b/(2a)$,

$$f=\sqrt{|b^2-4ac|}/(2a).$$

1.12. Вычислить значение функции

$$q=\begin{cases}1{,}7\ e^{-x}, & \text{если } c^2-x \geqslant 0;\\ 0{,}9\ e^{x}, & \text{если } c^2-x < 0.\end{cases}$$

1.13. Вычислить значение функции, заданной графически (рис. 1.4, *а—в*) по заданному значению аргумента x.

1.14. Вычислить значение функции

$$c = \begin{cases} \cdot\text{TRUE}\cdot, & \text{если } x > 0; \\ \cdot\text{FALSE}\cdot, & \text{если } x \leqslant 0. \end{cases}$$

1.15. Найти квадрат наибольшего из двух чисел a и b и вывести на печать признак $N=1$, если наибольшим является a, и $N=2$ — в противном случае.

1.16. Определить, попадает ли точка с координатами x_0, y_0 в круг радиусом r. Уравнение окружности $r^2=x^2+y^2$. Присвоить признаку $N=1$, если точка находится внутри круга, и $N=0$ — если вне круга.

1.17. Вычислить значение функции

$$z = \begin{cases} \ln x, & \text{если } x \geqslant 1; \\ 1, & \text{если } -1 < x < 1; \\ e^x, & \text{если } x \leqslant -1. \end{cases}$$

1.18. Определить, в каком квадранте находится точка с координатами x, y, и вывести на печать номер квадранта.

1.19. Округлить действительное положительное число x, меньшее 5, до ближайшего целого числа:

$$NX = \begin{cases} 0, & \text{если } x < 0{,}5; \\ 1, & \text{если } 0{,}5 \leqslant x < 1{,}5; \\ 2, & \text{если } 1{,}5 \leqslant x < 2{,}5; \\ 3, & \text{если } 2{,}5 \leqslant x < 3{,}5; \\ 4, & \text{если } 3{,}5 \leqslant x < 4{,}5; \\ 5, & \text{если } x \geqslant 4{,}5. \end{cases}$$

1.20. Определить, является ли значение целочисленной переменной x кратным трем. Если это имеет место, то вывести значение x на печать, в противном случае вывести на печать НЕТ.

1.3. АЛГОРИТМЫ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Часто при решении задач приходится многократно вычислять по одним и тем же математическим зависимостям при различных значениях входящих в них величин. Такие многократно повторяемые участки вычислительного процесса называются *циклами*. Использование циклов позволяет существенно сократить схему алгоритма и длину соответствующей ему программы. Различают циклы с заданным и неизвестным числом повторений. К последним относятся итерационные циклы, характеризующиеся последовательным приближением к искомому значению с заданной точностью.

● **1.21.** Вычислить и вывести на печать значения функции $y=a^3/(a^2+x^2)$ при x, изменяющемся от 0 до 3 с шагом 0, 1.

Это цикл с заданным количеством повторений, которое определяется как $n=](x_m-x_0)/h[+1$, где x_0 и x_m — соответственно начальное и конечное значение аргумента; h — шаг изменения аргумента.

Перед первым выполнением цикла необходимо задать начальное значение аргумента x, равное 0, а затем организовать 31 раз вычисление и печать значений функции y. При каждом новом выполнении цикла необходимо изменять аргу-

мент на шаг, равный 0,1. Чтобы процесс был конечным, необходимо задать условие окончания цикла. Таким образом, для организации цикла необходимо: задавать перед циклом начальное значение переменной, изменяющейся в цикле; изменять значение переменной перед каждым новым повторением цикла; проверять условие окончания цикла; управлять циклом, т. е. переходить к его началу, если он не закончен, или выходить из него по окончании. Последние три функции выполняются многократно.

Переменную, изменяющуюся в цикле, называют *параметром цикла*. В одном цикле может быть несколько параметров.

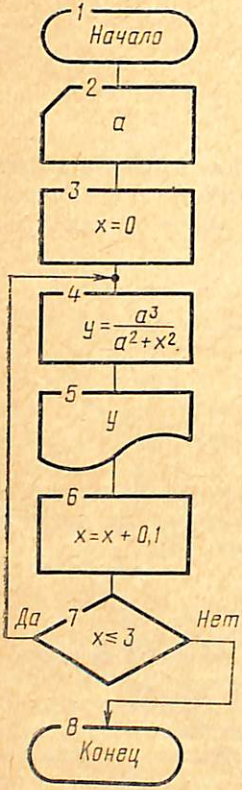


Рис. 1.5

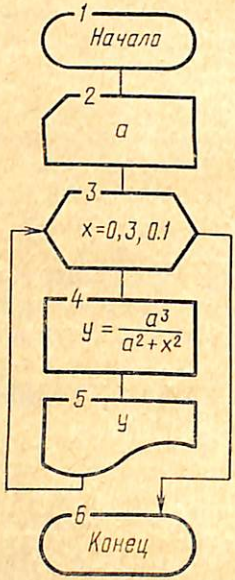


Рис. 1.6

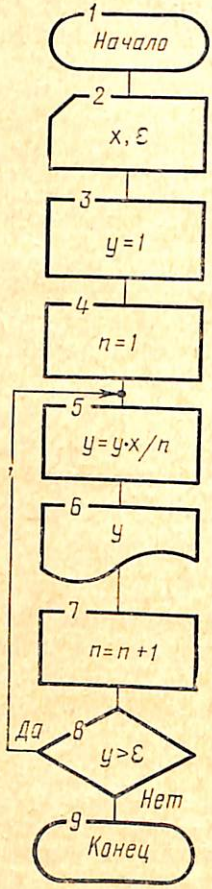


Рис. 1.7

Схема алгоритма приведена на рис. 1.5. На схеме блок *3* выполняет первую функцию, необходимую для организации цикла, блок *6* — вторую, блок *7* — третью и четвертую функции. Схема алгоритма получается во многих случаях более компактной и наглядной, если для ее построения использовать блок начала цикла, который выполняет все функции, необходимые для его организации (рис. 1.6).

● 1.22. Вычислить значения членов бесконечного ряда x, $\frac{x^2}{2!}$, $\frac{x^3}{3!}$, ..., $\frac{x^n}{n!}$, ... до члена $x^n/n! \leqslant \varepsilon$.

Здесь имеет место итерационный цикл, так как заранее не известно, при каком n выполняется условие $x^n/n! \leqslant \varepsilon$. Для итерационных циклов число повторений зависит от некоторого промежуточного или окончательного результата, а не от параметра цикла.

Сравнивая два соседних члена ряда, видим, что $y_n/y_{n-1} = x/n$. Поэтому для уменьшения времени счета при вычислении текущего члена ряда целесообразно использовать в цикле рекуррентную формулу $y_n = y_{n-1}x/n$.

Чтобы использовать эту формулу для вычисления значения первого члена ряда $y_1 = y_0$ ($x/1$), необходимо, чтобы заданное начальное значение y_0 было равно 1. Параметром, изменяющимся в этом цикле, будет номер n члена ряда. Тогда формула для вычисления значения текущего члена ряда будет иметь вид $y = yx/n$.

Схема алгоритма такого вычислительного процесса приведена на рис. 1.7. Если использовать для организации цикла блок начала цикла, то в нем надо указать в качестве последнего значения параметра цикла некоторое большое число, заведомо большее того n, при котором выполняется условие $y \leqslant \varepsilon$.

1.23. Вычислить значения функции $z = \sqrt{(x_i + a_i)}/2$, если x_i и a_i — элементы массивов, состоящих из 40 элементов каждый.

1.24. Вычислить и вывести на печать положительные значения функции $y = \sin nx - \cos n/x$ при $n = 1, 2, ..., 50$.

1.25. Вычислить значения функции $a_i = \begin{cases} a_i, & \text{если } a_i > 0; \\ 0, & \text{если } a_i \leqslant 0, \end{cases}$ если a_i — элементы массива $(a_1, a_2, ..., a_{25})$.

1.26. Записать в массив X, состоящий из 20 элементов, нули.

1.27. Вычислить значения функции $z = x^k/k^2$, большие a, если $k = 1, 2, 3, ...$

1.28. Для функции $z = x^k/k^2$ определить k, при котором z становится меньше a.

1.29. Вывести на печать положительные элементы массива $(x_1, x_2, ..., x_{60})$.

1.30. Вывести на печать первый отрицательный элемент массива $(a_1, a_2, ..., a_{50})$ и его порядковый номер, полагая, что в массиве есть хотя бы один отрицательный элемент.

1.31. Вывести на печать номера элементов массива $(y_1, y_2, ..., y_{100})$, удовлетворяющих условию $0 < y_i < 1$.

1.32. Вычислить $z = \sqrt[3]{a_i b_i c_i}/3$, где a_i, b_i и c_i — элементы массивов, состоящих из 20 элементов каждый.

1.33. Определить приближенное (с точностью 0,1) значение корня уравнения $x - \operatorname{arctg} x = \pi$. Изменяя значение x от 2 до 5 с шагом 0,1, вычислить функцию $y = x - \operatorname{arctg} x - \pi$. Изменение знака функции является признаком пересечения оси x. При $x = 2$ функция y отрицательная.

1.34. Вывести на печать номера точек, лежащих в круге радиусом r. Координаты точек заданы массивами $(x_1, x_2, ..., x_{100})$, $(y_1, y_2, ..., y_{100})$.

1.35. Определить с точностью до 0,2 точку пересечения функции $y=x-e^{-(ax)^2/2}$ с осью x при изменении аргумента x от b_0 до b_m с шагом 0,2. Сначала следует определить знак функции y при $x=b_0$. Изменение знака функции является признаком пересечения оси x.

1.36. Вычислить члены ряда $\frac{x^2}{2!}, -\frac{x^3}{3!}, ..., (-1)^n\frac{x^n}{n!}, ...$, модуль которых больше a.

1.37. В окружность радиусом r вписан многоугольник со стороной a_n. Сторона многоугольника с удвоенным числом сторон определяется по формуле $a_{2n}=\sqrt{2R^2-2R\sqrt{R^2-a_n^2/4}}$. Определить a_{128}, если известны r и a_4.

1.38. Вывести на печать положительные элементы главной диагонали матрицы **X** $(n\times n)$.

1.39. Рассчитать траекторию движения снаряда по формулам $x=v_xt$; $y=v_yt-gt^2/2$ при постоянных скоростях v_x, v_y. Время t изменяется от нуля с шагом Δt.

1.40. Вычислить $c=\cos nx=\cos((n-1)x)\cos x-\sin((n-1)x)\times\sin x$, если $\cos x=0{,}112$, $0<x\leqslant\pi/2$, $i=2, 3, ..., n$.

1.41. Определить количество цифр в целом числе n, если после деления числа n на 10^k раз в целой части числа будет нуль, где k — количество цифр в числе n.

1.42. Вывести на печать элементы массива $(x_1, x_2, ..., x_n)$, кратные трем $(n\leqslant 50)$.

1.4. ХАРАКТЕРНЫЕ ПРИЕМЫ АЛГОРИТМИЗАЦИИ ЗАДАЧ

Рассмотрим приемы, наиболее часто используемые при решении практических задач.

Вычисление в цикле с несколькими одновременно изменяющимися параметрами

В рассмотренных ранее примерах алгоритмов циклической структуры в цикле изменялся только один параметр. На практике часто встречаются задачи, в которых необходимо использовать несколько параметров цикла, изменяющихся одновременно. Цикл с несколькими одновременно изменяющимися параметрами организуется по схеме, аналогичной схеме организации цикла с одним параметром. Для остальных параметров перед циклом необходимо задавать их начальные значения, а внутри его вычислять текущие.

● **1.43.** Вычислить значения функции $z=xy_i/(x+y_i)$, если x изменяется одновременно с y_i от начального значения a с шагом h, y_i являются элементами массива $(y_1, y_2, ..., y_{20})$.

Здесь в цикле, выполняемом 20 раз, изменяются два параметра: простая переменная x и i — индекс переменной y.

Схема алгоритма решения этой задачи представлена на рис. 1.8, где блок *4* задает закон изменения параметра i от 1 до 20 с шагом 1, блок *3* — перед циклом начальное значение параметра x, а блок *7* вычисляет новое значение параметра x.

1.44. Вычислить значения функции $z=\sqrt{(x_i+a)/2}$, если x_i являются элементами массива $(x_1, x_2, ..., x_{30})$, a изменяется от 2 с шагом 0,5. Считать $x_i+a>0$.

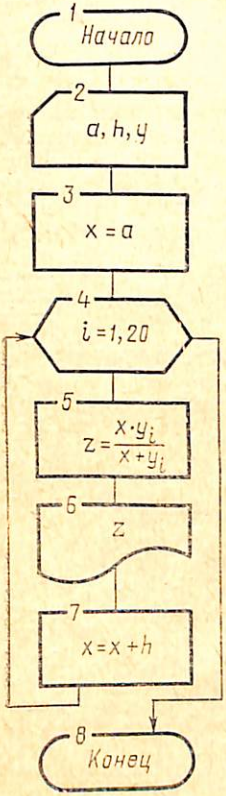


Рис. 1.8

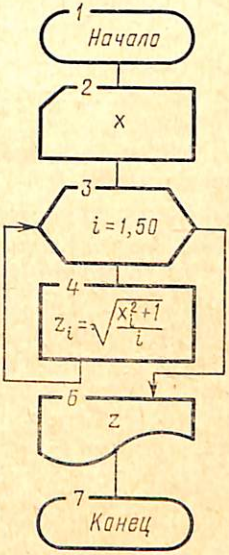


Рис. 1.9

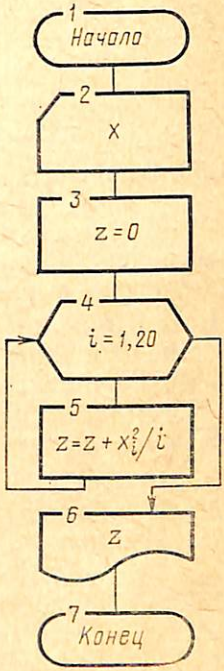


Рис. 1.10

1.45. Вычислить значения функции $z=(a+b+c_i)/i$, если a изменяется от 0 до 1 с шагом 0,1, b изменяется от 1 до 3 с шагом 0,2, c_i являются элементами массива $(c_1, c_2, ..., c_{11})$.

1.46. Вычислить значения функции $u=(x_i+y)/z_{2i-1}$, если x_i являются элементами массива $(x_1, x_2, ..., x_{50})$, z_i — массива $(z_1, z_2, ..., z_{100})$, а y изменяется от 1 с шагом 0,25.

1.47. Вычислить значения функции $z=x\sqrt{xy}$, где x изменяется от 1 с шагом 0,1 до 2, y изменяется от y_0 с шагом h. Считать $y>0$.

1.48. Вычислить и вывести на печать значения членов ряда

$$\frac{x+h}{3}, \quad \frac{x+2h}{5}, \quad \frac{x+3h}{7}, \ldots, \frac{x+nh}{2n+1}, \ldots, \frac{x+20h}{41}.$$

1.49. Вычислить и вывести на печать значения членов ряда

$$\frac{x_0+\Delta x}{n_0+3\Delta n}, \quad \frac{x_0+3\Delta x}{n_0+5\Delta n}, \ldots, \frac{x_0+(2n-1)\Delta x}{n_0+(2n+1)\Delta n}, \ldots .$$

Последнее значение знаменателя принять равным m.

Запоминание результатов

В рассмотренных выше задачах результатом вычислений было значение простой переменной, для записи которого в памяти ЭВМ выделяется одна ячейка. Если в процессе вычислений значение переменной изменяется, то в памяти ЭВМ после окончания вычислений остается лишь последнее значение результата. Чтобы записать все вычисленные значения, нужно выделить для их хранения необходимое количество ячеек памяти (массив), а текущий результат обозначить переменной с индексом.

● **1.50. Вычислить и запомнить значения функции** $z_i=\sqrt{(x_i^2+1)i}$, где x_i — элементы массива $(x_1, x_2, \ldots, x_{50})$.

Схема алгоритма, реализующего этот вычислительный процесс, представлена на рис. 1.9. В схеме блок печати стоит за циклом, так как на печать выводится весь массив Z.

1.51. Вычислить и запомнить значения функции $z=a\,e^{bx-cx^2}$ при изменении аргумента x от 0 до 2 с шагом 0,1.

1.52. Вычислить и запомнить значения функции $z_i=(x_i+y_i)/2$, где X и Y — массивы из 45 элементов каждый.

1.53. Вычислить и запомнить значения функции

$$y_i=\begin{cases} x_i, \text{ если } x_i>0; \\ 0, \text{ если } x_i=0; \\ -1, \text{ если } x_i<0, \end{cases}$$

где x_i — элементы массива из 20 элементов.

1.54. Переписать в массив Y элементы массива $(x_1, x_2, \ldots, x_{40})$ в обратном порядке.

1.55. Переписать положительные элементы массива $(x_1, x_2, \ldots, x_{100})$ подряд в массив Y.

1.56. Записать в массив N подряд номера положительных элементов массива $(a_1, a_2, \ldots, a_{80})$.

1.57. Записать в массив Y подряд десять первых положительных элементов массива $(x_1, x_2, \ldots, x_{100})$.

1.58. Вычислить значения функции $y=n\sin x-\cos nx$, если x изменяется от x_0 до x_m с шагом h. Записать в массив Z подряд значения функции, удовлетворяющие условию $0\leqslant y\leqslant 1$.

1.59. Записать подряд в массив B элементы массива $(a_1, a_2, \ldots, a_{75})$, имеющие четные индексы.

1.60. Записать подряд в массив B элементы массива $(a_1, a_2, \ldots, a_{100})$, стоящие на четных местах, а элементы, стоящие на нечетных местах, — в массив C.

1.61. Запомнить в массиве Z положительные значения y для монотонно убывающей функции $y=-x^3+ax^2+bx+c$, если x изменяется от 0 с шагом 0,1 до 10. Отрицательные значения функции не вычислять. Считать, что функция имеет хотя бы один отрицательный элемент.

1.62. Переписать положительные элементы массива (x_1, x_2, ..., x_{60}) в массив Y, а отрицательные — в массив Z. Элементы в массивах Y и Z располагать подряд.

1.63. Запомнить в массиве A значения n, при которых $z>0$ для знакочередующейся функции $z=\sin(nx+\varphi)$, а в массиве B — значения n, при которых $z\leqslant 0$ ($n=1, 2, 3, ..., 10$).

1.64. Переписать элементы главной диагонали матрицы A ($n\times n$) в одномерный массив B, считая $n\leqslant 30$.

Вычисление суммы и произведения

Если необходимо вычислить сумму значений некоторой функции $y=f(x)$ при различных значениях аргумента, целесообразно организовать цикл, в котором надо предусмотреть не только вычисление значений функции, но и накопление суммы путем прибавления полученных слагаемых к сумме всех предыдущих слагаемых. Формула, используемая для накопления суммы, имеет вид $z_n=z_{n-1}+y_n$. Поскольку надобности в запоминании значений всех слагаемых и промежуточных сумм нет, в качестве z и y нужно использовать простые переменные и накопление суммы вести в цикле по формуле $z=z+y$, где знак «=» означает присваивание значения. Если начальное значение z предварительно приравнять нулю, то после первого выполнения цикла значение z будет равно первому значению функции.

Аналогично накапливается и произведение с той лишь разницей, что для его накопления используется формула $z=zy$, а начальное значение произведения должно быть равно единице.

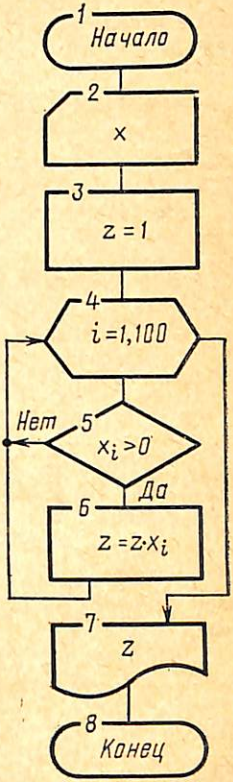


Рис. 1.11

● **1.65.** Вычислить значение функции $z=\sum_{i=1}^{20} x_i^2/i$, где x_i — элемент массива (x_1, x_2, ..., x_{20}).

Схема алгоритма решения этой задачи представлена на рис. 1.10. Блок *3*, задающий начальное значение суммы, стоит перед циклом, в котором накапливается эта сумма. Блок *5* вычисляет значение слагаемого и накапливает сумму. Поскольку результат решения этой задачи одно число, блок печати стоит за циклом и выполняется один раз.

● **1.66.** Вычислить произведение положительных элементов массива (x_1, x_2, ..., x_{100}).

Схема алгоритма решения задачи представлена на рис. 1.11. Значения сомножителей вычислять не требуется, поскольку они уже имеются в массиве. Однако прежде чем накапливать произведение, надо проверить, является ли сомножитель положительным (блок *5*). Блок *3* задает начальное значение произведения, равное единице. В одной из ветвей этого процесса стоит блок *6*, осуществляющий накопление произведения. При невыполнении условия $x_i>0$ никаких действий не предусматривается, а осуществляется переход к концу цикла.

1.67. Вычислить значения функции $z=\sum_{i=1}^{20}\sin x/\sqrt{1+x}$, , если x изменяется от 0 с шагом h одновременно с i.

1.68. Вычислить значения функции $z=\prod_{i=1}^{15}(n+i)/i$.

1.69. Вычислить сумму положительных элементов массива $(x_1, x_2, ..., x_{55})$.

1.70. Вычислить среднее арифметическое элементов массива $(a_1, a_2, ..., a_{60})$.

1.71. Вычислить сумму элементов массива $(a_1, a_2, ..., a_{78})$, стоящих на четных местах.

1.72. Вычислить сумму 20 первых элементов ряда

$$s=1+x+\frac{x^2}{2}+\cdots+\frac{x^{19}}{19}=1+\sum_{i=1}^{19}\frac{x^i}{i}.$$

Для определения x^i использовать прием накопления произведения, т. е. $x^i=x^{i-1}x$.

1.73. Вычислить среднее арифметическое отрицательных элементов массива $(c_1, c_2, ..., c_{30})$, полагая, что в массиве есть отрицательные значения.

1.74. Вычислить среднее арифметическое элементов s массива $(a_1, a_2, ..., a_{80})$, удовлетворяющих условию $1\leqslant a_i\leqslant 2$. Если таких элементов нет, то считать $s=0$.

1.75. Подсчитать количество положительных и количество отрицательных элементов массива $(x_1, x_2, ..., x_{75})$.

1.76. Вычислить значение функции $z=n!$

1.77. Вычислить сумму членов ряда $z=1+\frac{x^2}{2!}+\frac{x^4}{4!}+\ldots+\frac{x^{20}}{20!}=1+\sum_{i=1}^{10}\frac{x^{2i}}{(2i)!}$. Для вычисления значения члена ряда использовать рекуррентную формулу $y_n=y_{n-1}x^2/[2n(2n-1)]$.

1.78. Вычислить сумму членов ряда $z=1-x+\frac{x^3}{3!}-\frac{x^5}{5!}+\ldots+(-1)^n\frac{x^{2n-1}}{(2n-1)!}+\ldots-\frac{x^{21}}{21!}$. Для определения значения

члена ряда использовать формулу $y_{n+1}=y_n\frac{-x^2}{2n(2n+1)}$. Начальное значение $z=1-x$, а $y=-x$.

1.79. Вычислить значение функции $C_n^m=\frac{n!}{m!(n-m)!}=$

$$=\prod_{i=1}^{n-m}\frac{m+i}{i}.$$

1.80. Вычислить сумму элементов главной диагонали матрицы A(20×20).

1.81. Вычислить сумму элементов двух главных диагоналей матрицы A (10×10). Элементами i-й строки, лежащими на главных диагоналях, являются a_{ii} и $a_{i,11-i}$.

1.82. Вычислить сумму положительных и сумму отрицательных значений функции $z=\cos(nx+a)\sin(nx-a)$, где $n=1, 2, ..., 10$.

1.83. Вычислить среднее геометрическое элементов массива (y_1, y_2, ..., y_{25}), удовлетворяющих условию $y_i>a$, считая, что в массиве есть элементы, для которых выполняется это условие.

1.84. Вычислить среднее геометрическое положительных элементов массива (a_1, a_2, ..., a_{40}), имеющих четные индексы. Если таких элементов нет, то вывести на печать признак 0.

1.85. Вычислить приближенное значение функции Бесселя

$$J(x)=\sum_{k=0}^{100}\frac{(-1)^k\left(\frac{x}{2}\right)^{2k+n}}{k!\,(n+k)!}=\frac{\left(\frac{x}{2}\right)^n}{n!}+\sum_{k=1}^{100}\frac{(-1)^k\left(\frac{x}{2}\right)^{2k+n}}{k!\,(n+k)!}.$$

Для определения $n!$ использовать отдельный цикл, а слагаемого — рекуррентную формулу $y_k=y_{k-1}\frac{-(x/2)^2}{k(n+k)}$. В качестве начальных значений суммы и слагаемого использовать член ряда при $k=0$, равный $(x/2)^n/n!$

1.86. Вычислить размещение из n элементов по m, т. е. $a_n^m=n(n-1)(n-2)...(n-(m-1))$.

1.87. Подсчитать количество элементов целочисленного массива (x_1, x_2, ..., x_{60}), кратных трем.

1.88. Для целочисленного массива (a_1, a_2, ..., a_{75}) определить, является ли сумма его элементов четным числом, и вывести на печать ДА или НЕТ.

1.89. Подсчитать для массива (x_1, x_2, ..., x_{100}) количество элементов, ближайшим целым числом для которых является 1.

1.90. Вычислить сумму четных и сумму нечетных чисел натурального ряда от 1 до n.

Вычисление суммы членов бесконечного ряда

Задачи этого типа являются типичными задачами, использующими итерационный цикл, так как заранее не известно, при каком члене ряда будет достигнута требуемая точность. Выход из цикла организуется по условию достижения тре-

буемой точности. Для вычисления суммы членов ряда используется рассмотренный ранее прием накопления суммы.

● **1.91.** Вычислить сумму членов ряда $z=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots+$ $+(-1)^n\frac{x^{2n}}{(2n)!}+\cdots=1+\sum_{n=1}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}$ с точностью до члена ряда, меньшего ε. В целях уменьшения затрат времени на вычисление значения текущего члена ряда использовать рекуррентную формулу.

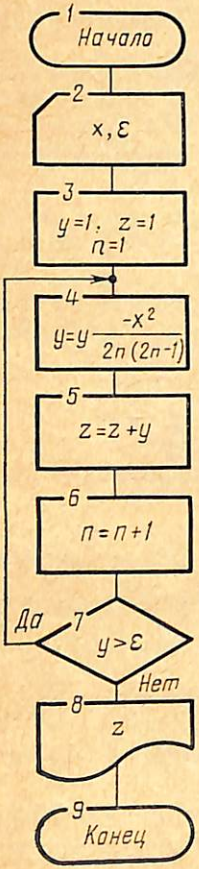


Рис. 1.12

Схема алгоритма решения этой задачи представлена на рис. 1.12.

Блок *3* задает начальное значение y, равное 1, начальное значение суммы, равное члену ряда с номером 0, так как вычислять его нет надобности, и начальное значение параметра цикла. В цикле блок *4* вычисляет значение текущего члена ряда, блок *5* накапливает сумму, блок *6* изменяет параметр цикла. Блок *7* проверяет условие повторения цикла и осуществляет переход к началу цикла, если $y>\varepsilon$, или выход из него в противном случае.

1.92. Вычислить сумму членов ряда

$$z=1+x+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}+\ldots$$

с точностью до члена ряда, меньшего ε.

1.93. Вычислить сумму членов ряда $z=1+\frac{mx}{(m+1)!}+\frac{m(m-1)}{(m+2)!}x^2+\frac{m(m-1)(m-2)}{(m+3)!}x^3+\ldots$ с точностью до члена, меньшего ε. Для определения текущего значения члена ряда использовать рекуррентную формулу $y_n=y_{n-1}\frac{x(m-n+1)}{n+m}$, где n — номер члена ряда. Начальное значение y принять равным $\frac{1}{m!}$.

1.94. Вычислить сумму членов ряда $y=\frac{1}{2\cdot3}+\frac{2}{3\cdot4}+\ldots+$ $+\frac{n}{(n+1)(n+2)}+\ldots$ с точностью до члена ряда, меньшего 10^{-4}.

1.95. Вычислить сумму членов ряда $z=\cos x+\frac{\cos 2x}{4}+$ $+\frac{\cos 3x}{9}+\ldots+\frac{\cos nx}{n^2}+\ldots$ с точностью до члена ряда, меньшего ε.

1.96. Вычислить сумму членов ряда $z=2\left[\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\right.$ $+\frac{(x-1)^5}{5(x+1)^5}+\ldots+\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}+\ldots$ с точностью до члена ряда, меньшего 10^{-6}.

1.97. Вычислить сумму членов ряда $y=\frac{\cos 2x}{1\cdot 3}+\frac{\cos 4x}{3\cdot 5}+$ $+\ldots+\frac{\cos 2nx}{(2n-1)(2n+1)}+\ldots$ с точностью до члена ряда, меньшего 10^{-4}.

1.98. Вычислить сумму членов ряда $y=1x^n+\frac{1}{2}x^{n-1}+\ldots+$ $+\frac{1}{n}x+\frac{1}{n+1}+\frac{1}{n+2}x^{-1}+\ldots$ с точностью до члена ряда, меньшего 10^{-6}. Для определения текущего члена ряда использовать рекуррентную формулу.

Вычисление полинома

Для вычисления полинома n-й степени $y=a_1x^n+a_2x^{n-1}+\ldots+a_nx+a_{n+1}$ удобно использовать формулу Горнера $y=(\ldots((a_1x+a_2)x+a_3)x+\ldots+a_n)x+a_{n+1}$. Если выражение, стоящее внутри скобок, обозначить y_i, то значение выражения в следующих скобках можно вычислить, используя рекуррентную формулу $y_{i+1}=y_ix+a_{i+1}$. Значение полинома y получается после повторения этого процесса в цикле n раз. Начальное значение y_1 целесообразно взять равным a_1, а цикл начинать с $i=2$. Если обозначить y простой переменной, то схема примет вид, показанный на рис. 1.13. Все коэффициенты полинома и свободный член, как правило, сводятся в массив, состоящий из $n+1$ элементов (n — порядок полинома). Если полином не содержит членов с некоторыми степенями x, то на соответствующем месте в массиве необходимо поместить коэффициент, равный 0.

1.99. Вычислить значение многочлена $y=2x^8-x^6+4x^5-5x^2+$ $+6x+1$, используя формулу Горнера.

Коэффициенты полинома удобно представить массивом (2; 0; -1; 4; 0; 0; -5; 6; 1). Порядок полинома n равен 8. Схема алгоритма будет аналогична схеме алгоритма, представленной на рис. 1.13.

1.100. Вычислить значение функции $\sin x=c_1x^9+c_2x^7+c_3x^5+$ $+c_4x^3+c_5x+x$, используя формулу Горнера $\sin x=((((c_1x^2+c_2)x^2+$ $+c_3)x^2+c_4)x^2+c_5)x+x$, где c_1, c_2, ..., c_5 — элементы массива.

1.101. Вычислить значение многочлена $z=2x^{12}-4{,}5x^{10}+x^9+$ $+3x^7-0{,}5x^4-x^2+1$, используя формулу Горнера.

1.102. Вычислить значение полинома $z=1x^8+2x^7+3x^6+4x^5+$ $+5x^4+6x^3+7x^2+8x+9$. Так как коэффициенты полинома — числа натурального ряда, то сводить их в массив не имеет смысла. Вычисление их целесообразно производить в процессе решения. Тогда формула для вычисления текущего значения полинома будет иметь вид $z_n=z_{n-1}x+n$.

1.103. Вычислить значение $s=(1+x)^8$, используя формулу Горнера $s=\left(\left(\ldots\left(\frac{1}{8}x+1\right)\frac{2}{7}x+1\right)\frac{3}{6}x+\ldots+1\right)8x+1$.

Множитель, на который умножается любая скобка, можно представить как $h_i=ix/(m-i+1)$, где $m=8$.

1.104. Вычислить сумму членов ряда $z=1+x+\frac{x^2}{2!}+\ldots+\frac{x^{12}}{12!}$, используя формулу Горнера $z=\left(\left(\left(\ldots\left(\frac{x}{12}+1\right)\frac{x}{11}+1\right)\frac{x}{10}+\ldots+1\right)\frac{x}{n}+\ldots+1\right)x+1$.

Нахождение наибольшего и наименьшего значения

Нахождение наибольшего и наименьшего значения функции $y=f(x)$ выполняется в цикле, в котором вычисляется текущее значение функции, и сравнивается с наибольшим или наименьшим из всех предыдущих значений этой функции. Если текущее значение функции окажется больше наибольшего из предыдущих значений, то его надо считать новым наибольшим значением. В противном случае

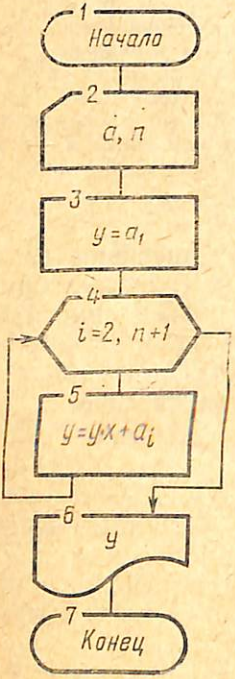


Рис. 1.13

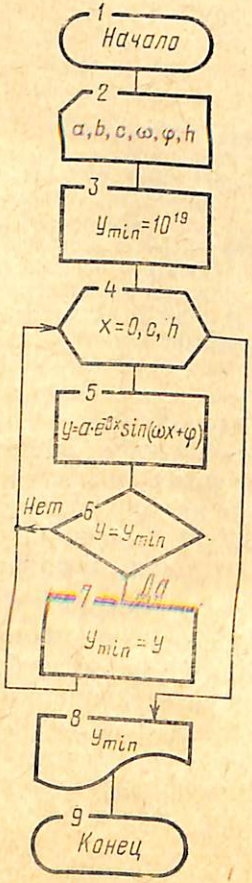


Рис. 1.14

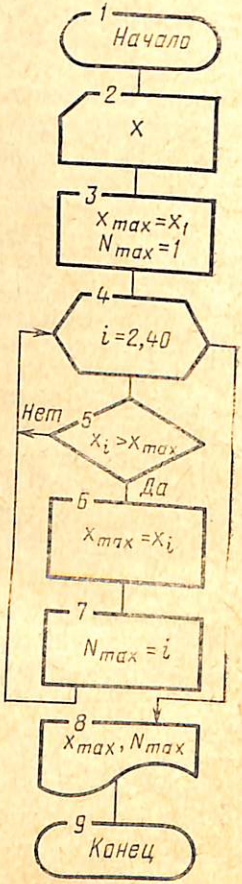


Рис. 1.15

наибольшее значение остается прежним. Сказанное можно описать условной математической формулой

$$y_{max} = \begin{cases} y_i, & \text{если } y_i > y_{max}; \\ y_{max}, & \text{если } y_i \leqslant y_{max} \end{cases} (i = 1,2,\ldots,n). \quad (1)$$

Аналогично, для наименьшего значения

$$y_{min} = \begin{cases} y_i, & \text{если } y_i < y_{min}; \\ y_{min}, & \text{если } y_i \geqslant y_{min}. \end{cases} \quad (2)$$

После первого выполнения цикла вычисляется y_1 и сравнивается с начальным значением y_{max} или y_{min}. После сравнения y_{max} или y_{min} принимает значение y_1. Тогда после вычисления y_2 будет находить наибольшее или наименьшее из этих первых двух значений функции. Необходимо в качестве начального значения y_{max} брать число порядка -10^{10}, чтобы наверняка выполнилось условие $y_1 > y_{max}$, а в качестве начального значения y_{min} — очень большое число, чтобы выполнилось условие $y_1 < y_{min}$.

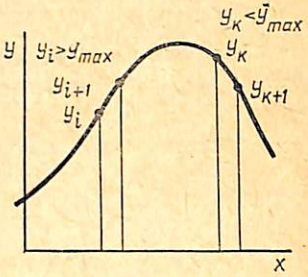


Рис. 1.16

Следует отметить, что здесь речь идет не о максимуме или минимуме функции, а о наибольшем или наименьшем из вычисленных значений функции. Это объясняется тем, что ЭВМ вычисляет дискретные значения функции и истинный максимум или минимум может находиться между ними.

● **1.105.** Найти наименьшее значение функции $y = ae^{-bx}\sin(\omega x + \varphi)$ в интервале изменения аргумента x от 0 до c с шагом h.

Схема алгоритма решения этой задачи представлена на рис. 1.14. Блок *3* задает перед циклом начальное значение $y_{min} = 10^{19}$. Блок *5* вычисляет текущее значение функции, а блоки *6* и *7* реализуют условную формулу (2).

● **1.106.** Найти наибольший элемент массива (x_1, x_2, ..., x_{40}) и его порядковый номер.

Здесь нет надобности вычислять сравниваемые значения, так как они уже имеются в массиве X. Поэтому в качестве начального значения x_{max} берется первый элемент массива. Поскольку сравнивать первый элемент массива с собой не имеет смысла, цикл выполняется начиная со второго элемента.

Схема алгоритма решения этой задачи представлена на рис. 1.15. Блок *3* перед циклом задает начальное значение $x_{max} = x_1$ и $N_{max} = 1$. В цикле блоки *5* и *6* реализуют условную формулу (1), блок *7* определяет номер наибольшего элемента массива.

● **1.107.** Найти экстремальное значение функции $y = |a|e^{bx+cx^2}$ при изменении аргумента x от 0 до 4 с шагом h. Функция такого вида имеет один экстремум. Если $c > 0$, то следует искать минимум, если $c < 0$, то — максимум.

Рассмотренный выше алгоритм нахождения наибольшего или наименьшего является универсальным, так как позволяет решить задачу даже в том случае, если функция не имеет экстремума или имеет несколько экстремумов. Если же функция имеет один экстремум, то затраты времени на решение такой задачи

можно существенно сократить, используя другой алгоритм. Пусть функция имеет один максимум. Тогда, вычисляя ее значения, лежащие до максимума, всегда будем получать новое значение функции, большее, чем наибольшее из всех предыдущих (рис. 1.16), т. е. всегда будет выполняться условие $y_i > y_{max}$. После максимума функция начинает убывать; следовательно, все последующие y_i будут меньше наибольшего. Условие $y_i < y_{max}$ можно использовать для выхода из цикла, так как среди последующих значений функции не может быть наибольшего. Этот процесс можно описать условной формулой

$$y_{max} = \begin{cases} y_i, \text{ если } y_i > y_{max}; \\ \text{выход из цикла, если } y_i \leqslant y_{max}. \end{cases}$$

Аналогично, для нахождения наименьшего

$$y_{min} = \begin{cases} y_i, \text{ если } y_i < y_{min}; \\ \text{выход из цикла, если } y_i \geqslant y_{min}. \end{cases}$$

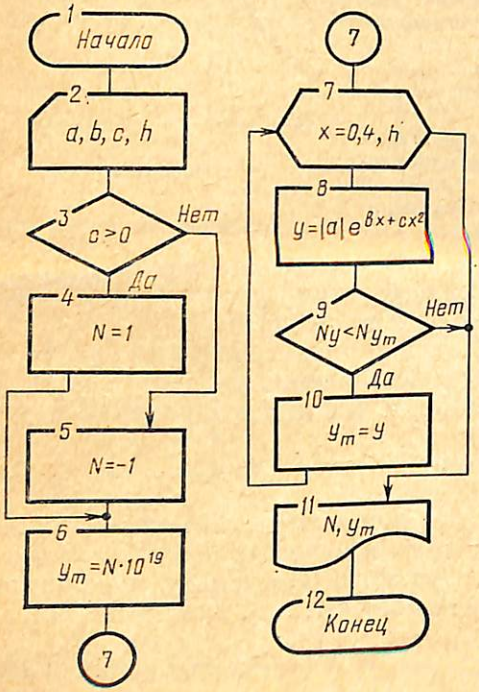


Рис. 1.17

Рассматриваемая функция **всегда положительна. В зависимости** от знака величины c она имеет или максимум, или минимум. Поэтому алгоритм должен начинаться с проверки знака величины c.

Схема алгоритма приведена на рис. 1.17. Блок *3* проверяет знак величины c, и если он положительный, то $N=1$ (функция имеет минимум), а если отрицательный, то $N=-1$ (функция имеет максимум). Блок *6* задает начальное значение y_m (экстремум). Если $N=1$, то $y_m=10^{19}$; если $N=-1$, то $y_m=-10^{19}$. Блок *9* в цикле проверяет условие $Ny<Ny_m$. Если $N=1$, то это условие аналогично условию $y<y_m$ (как и надо при нахождении минимума); если $N=-1$, то условию $y>y_m$ (как и надо при нахождении максимума). Если условие $Ny<Ny_m$ выполняется, то y считается новым экстремальным значением функции. В противном случае осуществляется выход из цикла к блоку *11*. Выход из цикла может осуществляться и естественным образом, когда цикл будет выполнен для всех значений аргумента x и условие выхода из цикла ни разу не выполнится. Блок *11* выдает на печать значение N — признак экстремума и y_m — экстремальное значение. Сокращение времени счета осуществляется за счет выхода из цикла, если условие $Ny<Ny_m$ не выполняется.

1.108. Найти номер (индекс) наименьшего элемента массива $(x_1, x_2, ..., x_{100})$.

1.109. Найти наибольшее значение (x_i+y_i) для массивов $(x_1, x_2, ..., x_{40})$ и $(y_1, y_2, ..., y_{40})$.

1.110. Найти наименьшее значение функции $y=ax^3+bx^2+cx+d$ и значение аргумента, при котором оно получено. Значение аргумента x изменяется от 0 до 10 с шагом 0,1.

1.111. Найти экстремумы функции $y=ax^3+bx^2+cx+d$, имеющей один максимум и один минимум при $a>0$ и $3ac-b^2<0$, если x изменяется от -10 до 10 с шагом 0,2 и максимума функция достигает при меньших значениях аргумента.

1.112. Для условия задачи 1.111 найти значения аргумента, при которых функция имеет наибольшее и наименьшее значения.

1.113. Найти экстремум функции $y=ae^{ax}-be^{-cx}$, если x изменяется от x_0 до x_m с шагом h. Функция имеет один экстремум. Для определения того, является этот экстремум максимумом или минимумом, можно использовать следующий способ. Вычислить два значения функции y_1 и y_2. Если $y_1>y_2$, то функция имеет максимум, в противном случае — минимум.

1.114. Записать $+1$ вместо максимального элемента массива $(x_1, x_2, ..., x_{50})$, а -1 вместо минимального.

1.115. Найти и записать вместо x_1 наибольший элемент, а вместо x_2 — наименьший для массива $(x_1, x_2, ..., x_{100})$.

1.116. Для массива $(a_1, a_2, ..., a_{80})$ вычислить наибольшее и наименьшее значения модуля разности между соседними элементами.

1.117. Для функции $y=ae^{-bx}\sin(\omega x+\varphi)$ найти первый максимум и первый минимум и значения аргумента, при которых они достигаются. Функция сначала достигает максимума. x изменяется от 0 до 6 с шагом 0,1.

1.118. Найти наибольший элемент главной диагонали матрицы **A** (20×20) и вывести на печать всю строку, в которой он находится.

1.119. Найти наименьший из положительных элементов массива $(x_1, x_2, ..., x_{40})$.

Уточнение корней уравнений

Аналитическое решение для многих алгебраических и трансцендентных уравнений получить не удается. Для решения таких уравнений используют приближенные итерационные методы (методы последовательных приближений). Решение уравнения производится определением грубого значения корня (например, графическим путем) и последующим его уточнением.

Уточнение значения корня уравнения рассматривается на примере метода итераций. Сущность метода заключается в том, что исходное уравнение представляется в виде $x=f(x)$.

Если в интервале между приближенным значением корня x_0 и корнем уравнения x выполняется условие $|f'(x)|<1$, то метод дает возможность вычислить значение корня с заданной точностью. Если это условие не выполняется, то надо перейти к обратной функции. Новое значение корня вычисляется через предыдущее по формуле $x_{i+1}=f(x_i)$. Повторяя этот процесс для x_1, x_2, x_3, ..., можно найти значение корня с заданной точностью, определяемой с помощью отношения $|x_i-x_{i+1}|\leqslant\varepsilon$.

● **1.120.** Вычислить наименьший положительный корень уравнения $x-\operatorname{tg} x=0$ с точностью $\varepsilon=10^{-5}$.

Преобразуем уравнение к виду $x=\operatorname{tg} x$. Для этого уравнения $|\operatorname{tg}' x|>1$, поэтому перейдем к обратной функции $x=\operatorname{Arctg} x$, где $\operatorname{Arctg} x=\operatorname{arctg} x+k\pi$ — неглавное значение функции. Будем считать все x_i (приближенные значения корня) простыми переменными. Для решения задачи требуется два смежных значения корня. Обозначим x_0 — предыдущее значение корня, а x_1 — последующее значение корня. Начальное значение наименьшего положительного корня $x_0=4{,}7$. Для вычисления нового значения корня будем использовать формулу $x_1=\operatorname{arctg} x_0+\pi$. Схема алгоритма решения этой задачи приведена на рис. 1.18. Блок *2* задает начальное (грубое) значение корня. Блок *3* вычисляет новое значение корня. Блок *4* проверяет достижение заданной точности. Если точность достигнута, то осуществляется выход из цикла, в противном случае выполняется блок *5* и переход к началу цикла. Блок *5* присваивает предыдущему значению корня только что вычисленное значение. На печать выводятся два смежных значения корня, так как корень уравнения лежит между ними.

1.121. Вычислить приближенное значение корня уравнения $x=\ln(x+2)$ с точностью $\varepsilon=10^{-4}$, используя метод итераций $(x_0=b)$.

1.122. Вычислить значение функции $y=1/\sqrt{x}$ по итерационной формуле $y_{i+1}=\frac{3}{2}y_i-\frac{1}{2}xy_i^3$ с точностью $\varepsilon=10^{-5}$, принять $y_0=2(2-\sqrt{2})$.

1.123. Определить корень уравнения $x-e^{-(ax)^2/2}=0$, используя метод итераций, с точностью $\varepsilon=10^{-5}$ $(x_0=b)$.

1.124. Решить систему уравнений $\begin{cases} x=x^2+y^2+0{,}05; \\ y=x^2-y^2+0{,}23 \end{cases}$ методом итераций с точностью $\varepsilon=10^{-4}$. Принять $x_0=0$, $y_0=0$.

1.5. АЛГОРИТМЫ СО СТРУКТУРОЙ ВЛОЖЕННЫХ ЦИКЛОВ

Любой цикл, содержащий внутри себя один или несколько других циклов, называется *вложенным*. Цикл, охватывающий другие циклы, называется *внешним*, а остальные циклы — *внутренними*. Правила организации как для внешнего, так и внутреннего циклов такие же, как и для простого цикла. Параметры этих циклов изменяются не одновременно, т. е. при одном значении параметра внешнего цикла параметр внутреннего цикла принимает по очереди все свои значения.

● **1.125.** Вычислить сумму положительных элементов каждой строки матрицы A (10×8).

Для вычисления суммы положительных элементов одной строки матрицы необходимо организовать цикл с целью перебора всех элементов строки, поэтому параметром этого цикла следует выбрать номер столбца j. Перед циклом нужно задать начальное значение суммы $s=0$. После окончания цикла результат необходимо вывести на печать. Если эти действия повторить во внешнем цикле, изменяя индекс строки i, то будут вычислены все 10 сумм. Схема алгоритма решения этой задачи приведена на рис. 1.19.

В рассмотренной задаче внешний цикл обязательно должен быть по i (индексу строки), так как в противном случае были бы вычислены суммы положительных элементов каждого столбца. Часто встречаются задачи, в которых не имеет значения, по какому параметру организовать внешний и внутренний цикл.

Все приемы программирования, изложенные выше, можно использовать и при организации вложенных циклов.

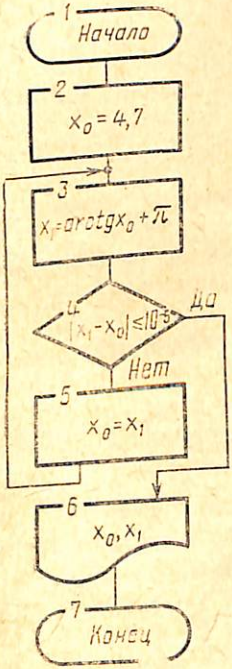


Рис. 1.18

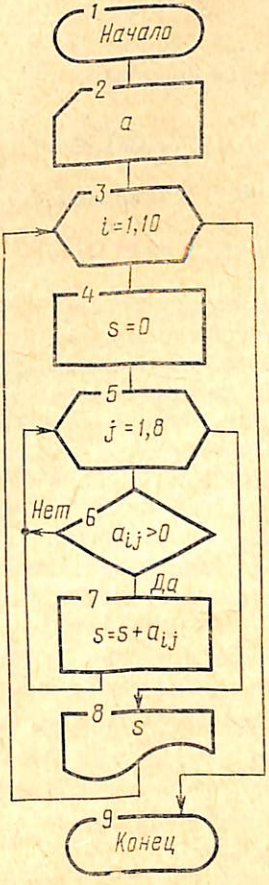


Рис. 1.19

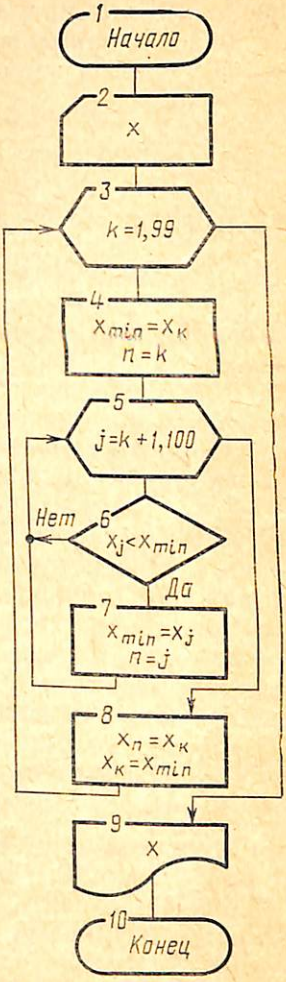


Рис. 1.20

● **1.126.** Упорядочить элементы массива (x_1, x_2, ..., x_{100}), расположив их в порядке возрастания в том же массиве. Для решения этой задачи требуется найти наименьший элемент. Поэтому перед внутренним циклом необходимо задать начальное значение наименьшего, а внутри цикла искать наименьшее и его порядковый номер.

После окончания цикла необходимо записать наименьший элемент в первую ячейку, а первый — в ячейку, где ранее был наименьший. Повторяя эти действия начиная со второго, затем третьего элемента и т. д., можно добиться того, что элементы массива будут упорядочены по возрастанию.

Схема алгоритма решения задачи представлена на рис. 1.20. Внешний цикл повторяется 99 раз, так как находить наименьший элемент из одного элемента x_{100} не имеет смысла. Блок *4* задает начальные значения $x_{\min}$ и номер n наименьшего элемента. Блоки *6, 7* находят наименьший элемент и его порядковый номер. Блок *8* записывает k-й элемент в n-ю ячейку и наименьший элемент в k-ю ячейку.

1.127. Вычислить значение функции $z=\sum_{i=1}^{10}\sum_{k=1}^{n}\frac{\sin^k(x_i)}{k}$,

где x_i заданы массивом $(x_1, x_2, ..., x_{10})$, $k=1, 2, ..., n$.

1.128. Вычислить значения функции $z_j=\prod_{i=1}^{20}(1+1/(e^i+x_j))$, где x_i заданы массивом $(x_1, x_2, ..., x_{40})$. Результаты запомнить в массиве Z.

1.129. Вычислить сумму элементов матрицы **A** (20×20), расположенных над главной диагональю.

1.130. Упорядочить элементы массива $(x_1, x_2, ..., x_{50})$, расположив их по убыванию в том же массиве.

1.131. Найти наибольшие элементы каждой строки матрицы **X** (10×20) и записать их в массив Y.

1.132. Найти среднее арифметическое положительных элементов каждого столбца матрицы **X** (15×25) при условии, что в каждом столбце есть хотя бы один положительный элемент.

1.133. Вычислить суммы элементов каждой строки матрицы **X** (20×20), определить наименьшее значение этих сумм и номер соответствующей строки.

1.134. Из матрицы **X** (10×15) построить матрицу *Y*, поменяв местами строки и столбцы.

1.135. Вычислить наибольшие значения функции $y_i=2e^{b_i x-5x^2}$, если b_i задано массивом $(b_1, b_2, ..., b_{20})$. Аргумент x изменяется от —2 до 2 с шагом 0,1. Все $y_{\max}$ запомнить в массиве C.

1.136. Определить количество положительных и отрицательных элементов матрицы **A** (10×15).

1.137. Определить количество положительных элементов каждого столбца матрицы **A** (10×20) и запомнить их в массиве M.

1.138. Упорядочить элементы массива $(x_1, x_2, ..., x_{60})$, расположив их по убыванию в массиве Y.

1.139. Найти наибольший элемент матрицы **A** (20×30) и номер строки и столбца, в которых он находится.

1.140. Найти наименьший элемент матрицы **X** (15×20) и записать нули в ту строку и столбец, где он находится.

1.141. Вычислить значение функции $z=\sum_{i=1}^{20} a_i \prod_{k=1}^{10} (a_i+b)/2$, где a_i заданы массивом $(a_1, a_2, ..., a_{20})$, b последовательно изменяется от 0 с шагом 0,1.

1.142. Найти три наибольших элемента массива $(a_1, a_2, ..., a_{30})$.

1.143. Перемножить матрицы **A** $(n\times m)$ и **B** $(m\times l)$. Элементы результирующей матрицы вычислять с помощью выражения $c_{ik}=\sum_{j=1}^{m} a_{ij}b_{jk}$.

1.144. Вычислить математическое ожидание $m_x=\frac{1}{100}\sum_{i=1}^{100} x_i$ и дисперсию $D_x=\frac{1}{100}\sum_{i=1}^{100}(x_i-m_x)^2$ случайных величин, записанных в массивах A, B и C по 100 элементов каждый. В целях экономии памяти все исходные данные хранить в массиве A.

1.145. Переписать первые элементы каждой строки матрицы **A** (15×25), большие c, в массив B. Если в строке нет элемента, большего c, то записать ноль в массив B.

1.146. Вычислить значения полинома $y_i=a_{i1}x^9+a_{i2}x^8+...+a_{i9}x+a_{i10}$ при различных значениях коэффициентов, используя формулу Горнера $y_i=(...(((a_{i1}x+a_{i2})x+a_{i3})x+...+a_{i9})x+a_{i10}$. Коэффициенты сведены в матрицу **A** (5×10).

● **1.147.** Определить с точностью $\varepsilon=0{,}01$ значение аргумента, при котором функция $y=ax-\ln x$ достигает минимума, при x, изменяющемся от 0,2 до 10.

Можно было бы решать эту задачу, взяв шаг изменения аргумента, равный 0,01. Однако это приведет к увеличению времени счета. Поэтому решение задачи разбивается на два этапа: 1) определение грубого значения минимума функции при большом шаге изменения аргумента, например 0,2; 2) повторение процесса в районе минимума при шаге изменения аргумента, равном 0,01.

Таким образом, при первом нахождении минимума шаг изменения аргумента равен 0,2, а его начальное значение $x_0=0{,}2$. При повторном нахождении минимума шаг равен 0,01, а $x_0=x_{\min}-0{,}2$.

Схема алгоритма решения задачи приведена на рис. 1.21.

Во внутреннем цикле осуществляется поиск наименьшего значения функции и значения аргумента, при котором оно достигается. Поскольку функция имеет один минимум, выход из цикла происходит при $y\geqslant y_{\min}$. В качестве параметра цикла взята некоторая переменная i, которая выполняет роль счетчика количества повторений цикла. После окончания внутреннего цикла проверяется условие $h=0{,}01$. Если выполнение условия имеет место, то осуществляется выход из внешнего цикла. В противном случае задаются новое

начальное значение переменной x, новый шаг h и внешний цикл повторяется еще один раз.

1.148. Найти значение аргумента x для функции $y=ae^{bx+cx^2}$, имеющей один максимум, при котором достигается максимум с точностью $\varepsilon=0{,}005$. Аргумент изменяется от -2 до 2 с шагом 0,1.

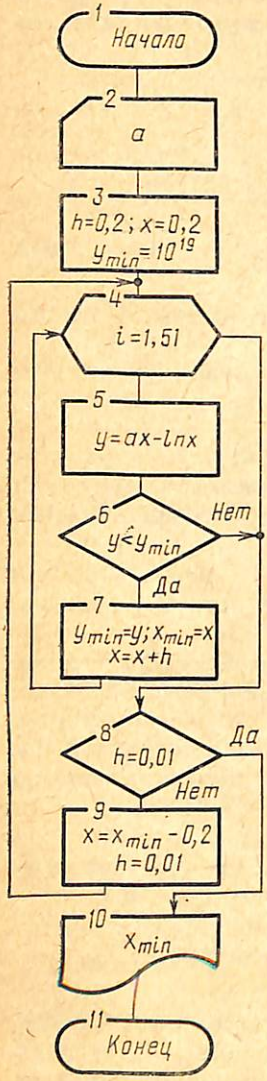


Рис. 1.21

1.149. Найти минимальные элементы каждой строки матрицы X (20×20) и поместить их на главную диагональ, а диагональные элементы записать на место минимальных.

1.150. Найти максимум функции $y=ax^2+bx^2+cx+d$ при изменении аргумента x от -10 до $+10$ с шагом 0,1 и минимум с шагом 0,01. Функция y имеет один максимум и один минимум при $a>0$ и $3ac-b^2<0$ и достигает максимума при меньших значениях аргумента.

1.151. Найти и запомнить в массиве Z начиная с первого максимума все максимумы и минимумы функции $y=ae^{-bx}\sin(\omega x+\varphi)$ при изменении аргумента x от 0 до 5 с шагом 0,1.

1.152. Вычислить значения функции $z=(a_i+bj)/c_k$; a_i, b_i, c_i заданы массивами из 10, 8 и 5 элементов соответственно.

2. ПРОГРАММИРОВАНИЕ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ ФОРТРАН

Ранее были перечислены этапы, предшествующие программированию на алгоритмическом языке, которые заканчивались составлением схемы алгоритма, описывающего вычислительный процесс. Эта схема универсальна, т. е. является основой для программирования на любом языке, в том числе и на языке ФОРТРАН. Основными конструктивными единицами языка программирования являются операторы, предписывающие, как и блоки схемы алгоритма, выполнение некоторых законченных действий. Задача этапа программирования состоит в том, чтобы записать операторы программы в той же последовательности, что и блоки в схеме алгоритма, не нарушив структуру связей. При этом надо учитывать специфику конкретного языка и правила записи его операторов.

2.1. ПРОСТЕЙШИЕ КОНСТРУКЦИИ ЯЗЫКА

К простейшим конструкциям языка относятся константы, переменные, функции и выражения. Для записи простейших конструкций и операторов языка используются следующие символы: все прописные латинские буквы, все арабские цифры, специальные знаки «+»; «—»; «*»; «/»; «(»; «)»; «.»; «,»; «=»; «␣»; «'»; «¤» «&»

Константы

В языке ФОРТРАН используются целые, действительные, логические, комплексные, шестнадцатиричные и текстовые (литеральные) константы. Все константы, кроме текстовой и комплексной, могут иметь стандартную (4 байта) и нестандартную длину.

Целые константы. *Целые константы* — последовательность цифр, перед которой может стоять знак плюс или минус. Например, 0; 1980; —121; +04; +15. Целые константы представляются в машине точно. Если они стандартной длины, то могут иметь значение, не превосходящее по абсолютной величине 2^{31}—1, если нестандартной длины, то не должны превосходить по абсолютной величине значения 2^{15}—1, т. е. 32767. Для хранения целой константы отводится 2 байта поля памяти.

Действительные константы. *Действительные константы* — числа, имеющие целую и дробную часть. Действительные константы представляются в машине приближенно. Допускается запись действительных констант в формах основной и с порядком. В первом случае константа имеет целую часть, отделенную точкой от дробной части. Нулевая целая или дробная часть константы может быть опущена. Например, 1.5; —17.07; 0.0; —0.4; —.76; 1.0; 4.6; +45. Основная константа стандартной длины состоит не более чем из семи цифр, а нестандартной длины (8 байтов) — не более чем из 16 цифр. Например, —17.049123456; —.1234567891011). Константы нестандартной длины более точно представляются в машине за счет большого количества разрядов, отводимых под константу. Во втором случае константа состоит из мантиссы, символа основания системы счисления и порядка. В качестве символа основания десятичной системы счисления записывается E для константы стандартной длины или D для константы нестандартной длины. Мантисса записывается по правилам записи констант, представленных или в основной форме, или целыми, порядок — по правилам записи целых **констант (порядок состоит не более чем из двух цифр).** Действительные константы по абсолютной величине не должны быть больше $\approx 7{,}2 \cdot 10^{75}$. Константа с порядком стандартной длины имеет мантиссу, содержащую не более семи цифр, а нестандартной длины — не более 16 цифр. Например, 0.5E—2; 17.E+04; 12E2; —.7E—12; +7.2E8; 1E0 — константы стандартной длины, 2.5D—17; 0.5D—2; 123.456789012D+25; 4D12 — константы нестандартной длины.

▲ **2.1.** Указать, какие из приведенных чисел — целые константы: 1) —3172; 2) 123456789; 3) 0; 4) +5000000017; 5) -2^{19}; 6) $0.573 \cdot 10^6$.

▲ **2.2.** Представить следующие числа в виде целых констант: 1) 00375; 2) $-2 \cdot 2^5$; 3) $0.645 \cdot 10^6$; 4) $2/5 \cdot 10^3$; 5) —0; 6) 10^8.

2.3. Представить числа: 1) 10^{-4}; 2) —297, 375256; 3) 1/3; 4) +9,12034001; 5) $295{,}3 \cdot 10^{-14}$; 6) $8971{,}0751 \cdot 10^{-6}$; 7) 1980; 8) —0,6; 9) $8765{,}8765 \cdot 10^{-3}$; 10) $0{,}257 \cdot 10^{-5}$; 11) $197813 \cdot 10^8$; 12) $2/3 \times \times 10^{12}$ в формах: а) основной стандартной длины; б) с порядком стандартной длины; в) с порядком нестандартной длины.

Комплексные константы. Комплексные константы записываются в виде пары действительных констант, разделенных запятой и заключенных в круглые скобки. Первая константа соответствует действительной, а вторая — мнимой части комплексного числа. Например, (2.5, 0.44), (1.5, 1.E—2), (0.0,1E2), (1.5, 2D—2). Комплексная константа имеет стандартную длину 8 байтов и нестандартную — 16 байтов.

Логические константы. Логические константы имеют одно из следующих двух значений: .TRUE. — истинно, .FALSE. — ложно. Стандартная длина логической константы 4 байта, нестандартная — 1 байт.

Текстовые константы. Текстовые константы — последовательность любых алфавитно-цифровых и специальных символов и могут иметь длину, не превышающую 255 символов; заключаются в апострофы, например 'КОРНИ УРАВНЕНИЯ' или записываются с использованием спецификации H, например 16H — КОРНИ УРАВНЕНИЯ.

Переменные

Для обозначения имен переменных, массивов, функций служат идентификаторы, представляющие собой комбинацию букв и цифр, начинающуюся обязательно с буквы. Количество символов в идентификаторе — от одного до шести. Например, X, YA, X2, SIGMA, S2C44, BETA5. Различают переменные простые и с индексами.

Простые переменные. Эти переменные в каждый момент времени имеют одно значение, которое хранится в одной ячейке. Простая переменная обозначается с помощью идентификатора.

Переменные с индексами. Переменные с индексами являются элементами массивов. Для хранения каждого элемента массива отводится своя ячейка. Переменная с индексами обозначается идентификатором массива, за которым в круглых скобках через запятую записываются от одного до семи индексов. Индекс представляется константой, переменной или выражением и должен быть целочисленным значением. Например, B(5) означает, что это пятый элемент массива B; A(I) означает i-й элемент массива A; X (5, 6) — элемент пятой строки, шестого столбца матрицы X; Z(3,2*N—1) — элемент третьей строки, $(2n—1)$-го столбца. Индекс не может быть числом, меньшим единицы.

2.4. Указать, какие комбинации символов являются обозначениями переменных: 1) R5A5; 2) 1B2ST; 3) Г2; 4) SUMMA; 5) σ; 6) OMEGA12; 7) X(1597); 8) N(2+A, SIN(X)); 9) α(1, 2); 10) D(I, 2*K, 5, 6, 1); 11) RAST(5, 3, 1); 12) B(0, 1, 0).

Описание типа переменных. Переменная может получать значение любой константы (целой, действительной, комплексной, логической или текстовой). Указание типа переменной осуществляется с помощью описаний. Существует три способа описания типа переменных.

Описание типа переменных по соглашению (умолчанию). Переменные, идентификаторы которых начинаются с букв I, J, K, L, M, N, считаются *переменными целого типа;* переменные, идентификаторы которых начинаются с других букв, — *переменными действительного типа.* Этот способ позволяет описывать только переменные целого и действительного типов стандартной длины (4 байта).

Для описания размерности массивов, тип которых определяется по соглашению, используется оператор DIMENSION. В нем указывается индентификатор массива и далее в круглых скобках максимальные значения индексов в виде целых чисел. Например, DIMENSION X(20), Y(5, 10) описывает одномерный массив X, индекс элементов которого изменяется от 1 до 20, двумерный массив Y, первый индекс которого (индекс номера строки) изменяется от 1 до 5, а второй индекс (индекс номера столбца) — от 1 до 10.

Описание типа переменных операторами явного описания. Эти операторы позволяют описывать переменные любого типа как стандартной, так и нестандартной длины, а также тип, длину и размерность массива. В качестве указателей типа используются ключевые слова REAL (действительный), INTEGER (целый), COMPLEX (комплексный), LOGICAL (логический), DOUBLE PRECISION (двойной точности).

Оператор явного описания имеет вид

$t*sa_1*s_1(k_1), ..., a_n*s_n(k_n)$,

где t — указатель типа; s — общий указатель длины в байтах, a_i — идентификатор переменной или массива, s_i — указатель длины a_i, k_i — размерность массива.

Если указатель длины s_i отсутствует, то длина определяется по s. Если s также отсутствует, то длина будет стандартной.

Оператор REAL*8X, Y(20), A*4(5,8) описывает действительные: переменную x длиной 8 байтов; массив Y, состоящий из 20 элементов длиной 8 байтов каждый, матрицу A, состоящую из 5 строк и 8 столбцов стандартной длины.

Оператор INTEGER X(40), Y*2, Z*2 (10) описывает целочисленные: массив X, состоящий из 40 элементов стандартной длины; переменную y и массив Z, состоящий из 10 элементов нестандартной длины (2 байта).

Операторы LOGICAL*1 C, N(5), T*4 и COMPLEX Q, D*16(5), E(8) описывают логические и комплексные переменные и массивы стандартной и нестандартной длины.

Описание типа переменных оператором IMPLICIT. Этот оператор описывает тип и длину переменных и массивов, идентификаторы которых начинаются с букв, указанных в нем.

Оператор IMPLICIT имеет вид

IMPLICIT $t_1*s_1c_1$, $t_2*s_2c_2$, ..., $t_n*s_nc_n$,

где t_i, s_i, c_i — соответственно указатели типа, длины, величин.

Оператор IMPLICIT INTEGER (A, D—F), REAL*8(R—T, Z) указывает, что переменные, начинающиеся с буквы A, а также с букв от D до F (т. е. D, E, F),— целые стандартной длины (так как s отсутствует), а переменные, начинающиеся с R, S, T и Z, — действительные нестандартные длины.

Следует иметь в виду, что описание переменных операторами явного описания отменяет описание оператором IMPLICIT и по соглашению.

▲ **2.5.** Описать переменные: 1) r и q как переменные целого типа стандартной длины, массив S целого типа из 10 элементов нестандартной длины, действительную переменную x и массив Y из 50 элементов нестандартной длины; 2) i, k как целочисленные переменные нестандартной длины, целочисленный массив N из 10 элементов стандартной длины, логическую стандартную переменную c; 3) k как действительную переменную стандартной длины, массив комплексных переменных из 40 элементов нестандартной длины.

▲ **2.6.** Описать переменные, используя все виды описаний: 1) i, j, k и l как переменные целого типа нестандартной длины, массив KOL целого типа из 15 элементов стандартной длины, a, b, c, d, e, f как действительные переменные нестандартной длины; 2) начинающиеся с букв от A до K как целочисленные нестандартной длины, массив D1 как действительный из 20 элементов нестандартной длины; 3) массивы X и Y как действительные из 30 элементов каждый, XS как действительную переменную стандартной длины, XSO как действительную переменную стандартной длины.

▲ **2.7.** Определить тип и длину переменных и массивов:

1) I, I1, L, N1, Q и массивов KOL, NOM, Q1, R, описанных в операторах

```
IMPLICIT INTEGER*2 (H—K, Q)
INTEGER I1, KOL (5), R (5,8)
DIMENSION NOM (8), Q1 (15)
```

2) LST, L, NOMER, N, N5A, X1, LIST, N5 и массивов X и LS, описанных в операторах

```
IMPLICIT REAL (L—P)
INTEGER*2 N5, X (50)
REAL*8 LST, LS (5), X1
```

3) N, INT, I, D1 и массивов D, E, IN, описанных в операторах

```
LOGICAL N*1. D (5)
REAL*8 INT, E (6)
DIMENSION IN (8)
```

Функции

Функция записывается в виде идентификатора функции и параметров, список которых указывается в круглых скобках. Например, $f(x_1, x_2, ..., x_n)$. Правила записи идентификаторов функций такие же, как и для переменных. За наиболее употребимыми функциями закреплены их имена. Некоторые из них приведены в табл. 2.1. Значением этих функций может быть константа целого, действительного или комплексного типа стандартной или нестандартной длины. Аргумент может быть константой, переменной, функцией или выражением.

Выражения

Различают выражения арифметические и логические.

Арифметическое выражение. Оно записывается с помощью констант, переменных, функций, знаков арифметических операций и круглых скобок. Результатом вычисления значения арифметического выражения является числовая константа (целая, действительная или комплексная).

Таблица 2.1

Функция	Математическое обозначение функции	Запись функции в языке ФОРТРАН	Функция	Математическое обозначение функции	Запись функции в языке ФОРТРАН
Синус	$\sin x$	SIN(X)	Экспонента	e^x	EXP(X)
Косинус	$\cos x$	COS(X)	Логарифм натуральный	$\ln x$	ALOG(X)
Тангенс	$\mathrm{tg}\, x$	TAN(X)			
Арксинус	$\arcsin x$	ARSIN(X)	Корень квадратный	$\sqrt{x}$	SQRT(X)
Арккосинус	$\arccos x$	ARCOS(X)			
Арктангенс	$\mathrm{arctg}\, x$	ATAN(X)	Модуль	$\lvert x \rvert$	ABS(X)

Примечание. Все функции действительные стандартной длины. Имя действительной функции нестандартной длины начинается с буквы D, а комплексной — с буквы C. Например, DSIN, DLOG, CEXP.

Таблица 2.2

X \ Y	I*2	I*4	R*4	R*8	C*8	C*16
I*2	I*2	I*4	R*4	R*8	C*8	C*16
I*4	I*4	I*4	R*4	R*8	C*8	C*16
R*4	R*4	R*4	R*4	R*8	C*8	C*16
R*8	R*8	R*8	R*8	R*8	C*16	C*16
C*8	C*8	C*8	C*8	C*16	C*8	C*16
C*16	C*16	C*16	C*16	C*16	C*16	C*16

Примечание. I — целая величина, R — действительная, а C — комплексная. После * указывается длина в байтах.

Порядок выполнения операций в выражении задается скобками. При отсутствии скобок операции выполняются в порядке старшинства: вычисление функции, возведение в степень, умножение или деление, сложение или вычитание. Операции одного порядка выполняются последовательно слева направо. Операция извлечения корня заменяется возведением в дробную степень.

Тип и длина результата арифметической операции в зависимости от участвующих в ней операндов приведены в табл. 2.2.

Следует иметь в виду, что комплексную константу можно возводить только в целую степень и комплексная константа не может быть показателем степени.

▲ 2.8. Записать следующие арифметические выражения:

1) $\frac{x}{1+x^2/(2y)}$; 2) $\frac{a+bx}{cd}$; 3) $e^x-\sin^2 x$; 4) $1+\operatorname{arctg}\frac{x}{1+\sqrt{x}}$;

5) x^{y^z}; 6) $\frac{a\sqrt{x}-b^2\sqrt[3]{x}}{\sqrt[5]{x+0{,}5}}$; 7) $ax^3-bx^2+a\cos^2|x|$;

8) $\frac{\sin x^3-\cos x^3}{2x}-10^{-3}$; 9) $\frac{x^m-a^m}{\sqrt{ax}}$; 10) $e^{-ax}\sin(\omega t+\varphi)$;

11) $\frac{\ln|x-a|}{x^2-ax+a^2}$; 12) $x^{1/3}-2y^{2/5}+\varepsilon$.

▲ 2.9. Определить тип и длину результата следующих арифметических выражений: 1) $\cos(2n)+kx$ (все переменные описаны по соглашению); 2) $\sqrt[3]{ax^2-nx+1}$ (все переменные описаны по соглашению); 3) $k^2+n+0{,}5$ (k и n — целые переменные нестандартной длины); 4) $(i+10)^5-n^2$ (i — целая переменная нестандартной длины, n — целая переменная стандартной длины); 5) $\frac{x_1+x_2}{2a}$ (x_1 и x_2 — комплексные переменные стандартной длины, a — действительная переменная нестандартной длины); 6) $(a^2-b^2)/d$ (a и b — действительные переменные стандартной длины, d — действительная переменная нестандартной длины).

Логическое выражение. Простейшее логическое выражение — отношение вида $a\alpha b$, где α — знак операции отношения: .GT.—>; .GE.—≥; .LT.—<; .LE.—≤, .EQ.—=; .NE.—≠, a и b — арифметические выражения целого или действительного типа. Более сложные логические выражения строятся с помощью знаков логических операций: .NOT.—НЕ; .AND.—И; .OR.—ИЛИ.

Логическое выражение может принимать значения или .TRUE., или .FALSE.. Значения результатов логических операций приведены в табл. 2.3.

Порядок приоритета логических операций следующий: сначала вычисляется значение отношений, затем операции .NOT., далее .AND. и, наконец, операция .OR..

▲ 2.10. Определить значения следующих логических выражений: 1) X.GT.0.AND.Y.GT.0 при X=5, Y=—1; 2) X.GT.0.OR.Y.GT.0 при X=0, Y=2; 3) X.EQ.0.OR..NOT.Y.NE.O при X=1, Y=—2; 4) .NOT. A.OR.B.AND.D.GT.C при A=.FALSE., B=.TRUE., D=5.5, C=3.8; 5) .NOT.X.AND.Y.OR.Z.AND..TRUE. при X=.FALSE., Y=.TRUE., Z=.FALSE.; 6) (.NOT.X.AND.Y.OR.Z).AND..TRUE., при X= =.FALSE., Y=.TRUE., Z=.FALSE.

Таблица 2.3

A	.TRUE.	.TRUE.	.FALSE.	.FALSE.
B	.TRUE.	.FALSE.	.TRUE.	.FALSE.
.NOT.A	.FALSE.	.FALSE.	.TRUE.	.TRUE.
A.AND.B	.TRUE.	.FALSE.	.FALSE.	.FALSE.
A.OR.B	.TRUE.	.TRUE.	.TRUE.	.FALSE.

▲ **2.11.** 1. Записать логические выражения, определяющие условия, при которых можно вычислять следующие арифметические выражения: а) $z=1/(x^3y^3)$; б) $z=a/(x\ln y)$;

в) $z=\sqrt{x}/(\ln y+x)$; г) $x=(b-\sqrt{b^2-4ac})/(2a)$;

д) $a=\arcsin x+\arccos y$; е) $z=\ln y/(x-y)$.

2. Записать условия кратности трем для целочисленной переменной n.

3. Записать условия кратности пяти одновременно двух целочисленных переменных a и b.

2.2. ВВОД И ВЫВОД ДАННЫХ

Ввод и вывод значений простых переменных

Для ввода и вывода данных используются операторы READ и WRITE совместно с оператором FORMAT. Общий вид операторов:

READ (m, n) s

WRITE (m, n) s

n FORMAT (c)

где n — метка оператора FORMAT, s — список переменных для ввода — вывода, c — список спецификаций, m — номер устройства ввода — вывода.

В DOC ЕС для устройства ввода с перфокарт $m=1$, а для устройства печати $m=3$, а в ОС ЕС соответственно $m=5$ и $m=6$.

Список спецификаций определяет формат данных и их расположение на карте при вводе или на бумажной ленте при печати. Наиболее употребимы спецификации:

aIw — для ввода — вывода значений переменных целого типа;

aF$w.d$ — для ввода — вывода значений действительных переменных, представленных в основной форме;

aE$w.d$ и aD$w.d$ — для ввода — вывода действительных переменных в форме с порядком стандартной и нестандартной длины;

aX — для пропуска позиций при вводе и выводе;

aH<текстовая константа>;

литериал — для вывода на печать текстовых констант;

aLw — для ввода — вывода логических констант, где w — общее количество позиций, отводимых под константу на перфокарте или бумажной ленте;

d — количество позиций, отводимых под дробную часть константы;

a — коэффициент повторения спецификации.

Например, 3I5 — надо ввести или вывести три константы целого типа, под каждую из которых отводится пять позиций; 2F6.2 — вводятся или выводятся две действительные константы в основной форме, каждая из которых занимает вместе со знаком числа и десятичной точкой шесть позиций, в том числе две позиции для дробной части; 6E12.5 — вводятся или выводятся шесть действительных констант в форме с порядком, каждая из которых занимает 12 позиций (знаки числа и порядка, две цифры порядка, буква E, десятичная точка, цифра в целой части мантиссы, пять цифр в дробной части мантиссы) (очевидно, $w \geqslant d+7$); 5X — пропустить пять колонок при вводе информации с перфокарты или пять позиций при печати.

'␣РЕШЕНИЕ␣УРАВНЕНИЯ' или 18H␣РЕШЕНИЕ␣УРАВНЕНИЯ означает вывод на печать 18 символов;

2L4 — вводятся или выводятся две логические константы, занимающие четыре позиции (при выводе будет печататься␣␣␣T для константы, имеющей значение. TRUE., и␣␣␣F при значении .FALSE.).

При выводе на печать первый символ в строке является управляющим и не печатается. Он может иметь следующие значения: ␣ (пробел) — пропуск одной строки, 0 — пропуск двух строк, 1 — печать с новой страницы, + — печать в той же строке.

В операторе FORMAT может использоваться косая черта, означающая переход на новую карту при вводе или на новую строку при выводе. Например,

Оператор FORMAT (I3/5F4.1) означает, что на первой карте (строке) в трех позициях располагается целая константа, а на второй карте (строке) — пять констант в основной форме.

В *операторе* FORMAT (I3/(5E14.7)) внутренние скобки означают, что начиная со второй карты (строки) все остальные карты (строки) содержат константы, соответствующие спецификации 5E14.7.

Для ввода значений переменных: I=170, K=—15, A=—23.065, B=—0.5E—12 операторы ввода и FORMAT должны быть следующими:

```
READ (5,7) I, K, A, B
7 FORMAT (2I3, F7.3, E8.1)
```

При печати значений этих же переменных операторы вывода и FORMAT с учетом пробелов между печатаемыми числами будут иметь вид

```
WRITE (6,8) I, K, A, B
8 FORMAT (1H␣ , 2I4, F8.3, E9.1)
```

При этом числа на бумаге будут располагаться так:␣170␣—15␣—23.065␣—0.5E—12

Если необходимо отпечатать значения переменных вместе с их наименованиями, то используют спецификации H или литерал.

Тогда оператор FORMAT имеет вид

```
8 FORMAT (3H␣I=, I3, 3H␣K=, I3, '␣A=', F7.3',␣ B=', E8.1)
```

При этом данные будут отпечатаны в следующем виде:

I=170␣K=—15␣A=—23.065␣B=—0.5E—12

▲ **2.12.** Записать операторы ввода и вывода для следующих переменных: 1) NAME=1, KOL=120, A=—172.04; 2) X=—25.004, Y=170.725, Z=0.17E—3; 3) A=70.0, B=—1.2, C=—7.3 (вывод переменных осуществить вместе с их наименованиями); 4) E=0.573E—12, F=121.001D—25, D=0.235D17; 5) A=—2., B=7.5, C=1E—6. Перед выводом на печать отпечатать заголовок ИСХОДНЫЕ ДАННЫЕ.

▲ **2.13.** Записать 1) операторы ввода X=0.9879E—2, Y=—197.E3, используя спецификацию F; 2) операторы ввода Z=—0.12597E2,

W=0.63574D8, используя спецификации E и D; 3) операторы ввода K=121, N=—8, X=0.50E6, используя спецификации I и E; 4) операторы вывода для печати заголовка РЕЗУЛЬТАТЫ СЧЕТА; 5) операторы вывода для печати заголовка КОРНИ УРАВНЕНИЯ, а в следующей строке X1=0.66666 X2=—0.33333; 6) операторы вывода для печати заголовка таблицы АРГУМЕНТ__X ___ __ФУНКЦИЯ__Y.

Ввод и вывод значений элементов массивов

Существуют три способа организации ввода и вывода массивов.

1. Массив вводится или выводится целиком. В этом случае в операторе ввода или вывода указывается только имя массива:

```
      DIMENSION X (20)
      READ (5,2) X
2     FORMAT (10 F8.3)
      WRITE (6,3) X
3     FORMAT (10 F12.3)
```

2. Ввод и вывод элементов массива в цикле:

```
      DIMENSION X (20)
      DO 2 I=1, 20
      READ (5,1) X (I)
1     FORMAT (F 8.3)
2     WRITE (6,3) X (I)
3     FORMAT (F 12.3)
```

3. Использование автоматических индексаций:

```
      READ (5,1) (X(I), I=1,20)
1     FORMAT (10F8.3)
      WRITE (6,2) (X(I), I=1,20)
2     FORMAT (10F12.3)
```

Последние два способа позволяют осуществлять ввод и вывод части массивов:

```
      DIMENSION X (20)
      READ (5,1) (X(I), I=1,15)
1     FORMAT (10 F8.3)
```

Эти способы применимы также и для двумерных, и многомерных массивов. Например, ввод матрицы с использованием автоматических индексаций имеет вид:

```
      DIMENSION X (5, 10)
      READ (5,1) ((X(I, J), I=1,5), J=1,10)
1     FORMAT (10F8.3)
```

Первым изменяется индекс, заключенный во внутренние скобки. При такой записи оператора ввода элементы матрицы будут вводиться по столбцам.

При выводе

```
      DIMENSION X(10, 8)
```

. . . . . .

```
      WRITE (6,2) ((X (I, J), J=1,8), I=1,10)
2     FORMAT (8E12.4)
```

матрица будет отпечатана в общепринятом виде по восемь элементов в каждой строке:

```
      DIMENSION X (20, 30)
      READ (5,1) N, M, ((X, (I, J), I=1, N), J=1, M)
1     FORMAT (2I2/(10F8.3))
```

Пример показывает ввод матрицы $n \times m$ ($n \leqslant 20$, $m \leqslant 30$): сначала вводятся значения n и m по спецификации 12. Косая черта в операторе FORMAT означает, что перед ней указывается расположение чисел на первой карте, а после нее на второй. Внутренние скобки для 10F8.3 означают, что не только на второй, но и на всех последующих картах располагается по 10 чисел в спецификации F. Аналогично назначение косой черты и при выводе:

```
      WRITE (6,5) K, N, C, (T(K), K=1,120)
5     FORMAT (1H0, 2I6/E15.5/(10E12.4))
```

2.14. Записать операторы ввода массива C, состоящего из 200 элементов. Под каждое число отвести по пять позиций, предусмотрев в дробной части одну цифру.

▲ **2.15.** Записать операторы ввода массивов X и Y, состоящих из 100 элементов каждый. Под каждое число отвести восемь позиций, предусмотрев в дробной части три цифры.

▲ **2.16.** Записать операторы ввода значений переменных a, b, c, под которые отвести соответственно пять, шесть и семь позиций, и массива D, состоящего из 80 элементов, занимающих по шесть позиций каждый. Для дробной части всех чисел отвести две позиции.

▲ **2.17.** Записать операторы ввода значений первых m элементов, используя спецификацию F6.2, в массив D, состоящий из 120 элементов.

▲ **2.18.** Записать операторы ввода матрицы A ($n \times m$), где $n \leqslant 30$, $m \leqslant 30$, используя спецификацию F5.1. Элементы матрицы расположить по столбцам.

▲ **2.19.** Записать операторы ввода значений переменных a и b, используя спецификацию F6.3, и матрицы **X** ($n \times n$) ($n \leqslant 100$), используя спецификацию F4.2. Элементы матрицы расположить по строкам.

▲ **2.20.** Записать операторы ввода целочисленного массива M, состоящего из k элементов ($k \leqslant 120$). Каждый элемент массива должен занимать три позиции.

▲ **2.21.** Записать операторы ввода массивов NOM и X, состоящих из n элементов ($n \leqslant 100$), занимающих по пять позиций каждый. Элементы массивов расположены парами: NOM_1, X_1, NOM_2, X_2, и т. д.

▲ **2.22.** Решить задачу 2.21, если элементы массивов расположены в таком порядке: сначала элементы массива NOM, затем элементы массива X.

▲ **2.23.** Записать оператор печати массива Z, состоящего из 50 элементов. В каждой строке отпечатать 10 чисел, используя спецификацию F12.4.

▲ **2.24.** Записать оператор печати массивов X и Y, состоящих из 20 элементов каждый. Сначала отпечатать элементы массива X, а затем массива Y, используя спецификацию E12.4.

▲ **2.25.** Отпечатать значения аргумента и функции в два столбца, используя спецификации F6.2 и E12.3 соответственно, если значения аргумента сведены в массив X, а функции — в массив Y. Оба массива состоят из 10 элементов.

▲ **2.26.** Записать оператор печати значения переменной m на одной строке по спецификации I5 и массива Y, состоящего из m элементов, по восемь элементов на каждой строке, используя спецификацию E15.5.

▲ **2.27.** Записать оператор печати массивов A, B, C, состоящих из 20 элементов каждый. В строке печатать порядковый номер элемента и по одному элементу каждого массива по спецификации E20.5.

▲ **2.28.** Записать оператор печати матрицы **A**, состоящей из 10 строк и 8 столбцов, в общепринятом виде, используя спецификацию E13.4.

▲ **2.29.** Записать оператор печати матрицы **A**, состоящей из 30 строк и 30 столбцов. Элементы располагать по строкам, используя спецификацию E12.3.

▲ **2.30.** Записать оператор печати матрицы **Z**, состоящей из 15 строк и 8 столбцов, в общепринятом виде, используя спецификацию E12.3. Слева перед каждой строкой печатать номер строки.

▲ **2.31.** Решить задачу 2.30, выводя на печать не только номера строк, но и столбцов. Номер столбца печатать над серединой элемента. Номера столбцов сведены в массив NOM.

▲ **2.32.** Вывести на печать левую треугольную матрицу, используя спецификацию F8.2, если дана матрица **X** (10×10).

▲ **2.33.** Решить задачу 2.32, выводя на печать правую треугольную матрицу.

▲ **2.34.** Вывести на печать элементы главной диагонали матрицы A (10×10), используя спецификацию E12.3.

▲ **2.35.** Записать оператор печати матрицы **C**, состоящей из восьми строк и пяти столбцов, в общепринятом виде, используя спецификацию E20.5. Перед матрицей отпечатать заголовок ЗНАЧЕНИЯ МАТРИЦЫ **C**.

2.3. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ЛИНЕЙНОЙ СТРУКТУРЫ

Все операторы языка ФОРТРАН подразделяются на выполняемые, используемые для указания действий и порядка их выполнения, и невыполняемые, служащие для описания величин, формы представления данных при вводе — выводе и т. д. Для записи линейных программ чаще используются: выполняемые операторы — присваивания, ввода, вывода, оператор останова (STOP) и невыполняемые операторы FORMAT и END.

Оператор присваивания служит для вычисления значения арифметического или логического выражения и присваивания этого значения имени переменной. Общий вид оператора

$v=a$,

где v — имя переменной, значение которой вычисляется; a — арифметические или логические выражения.

Если выражение a арифметическое, то и переменная v должна принимать числовое значение. Если выражение a логическое, то и переменная v должна быть логической. Результат вычисления выражения преобразуется к типу и длине, соответствующим переменной v. Например, X=N+1. Значение выражения N+1 — целое число. Переменная X по соглашению является действительной, поэтому целое число будет сначала преобразовано в действительную форму и затем присвоено переменной X. Если z описана как целая переменная нестандартной длины, то какой бы тип и длину ни имел результат вычисления выражения правой части оператора Z=2*X*(sin(N*X)+K), ему будет присвоена целая константа нестандартной длины. Преобразование действительной константы в целую осуществляется отбрасыванием дробной части числа.

Если результат вычисления выражения является комплексной константой, а переменная в левой части оператора — целая или действительная, то коэффициент при мнимой единице отбрасывается. Если переменная в левой части оператора комплексная, а выражение имеет действительное значение, то переменной будет присвоено значение, состоящее из действительной части комплексной переменной, а мнимая часть получит значение 0.

▲ **2.36.** Записать операторы присваивания для выражений

1) $c=\dfrac{ax+b^3}{\sqrt{(a-b)^3}}$; 2) $r=\dfrac{x}{a}-\dfrac{1}{\pi}\ln(a+be^{px})$; 3) $t=\dfrac{1}{x^2}\left(\dfrac{y}{10^2}\right)^x$;

4) $\alpha=2\sin^3(\pi n-x)$; 5) $p(x)=((((a_5+a_4)x+a_3)x+a_2)x+a_1)x+a_0$; где a_i — элементы массива; 6) $e=a>b$; 7) $f=a^2+1\geqslant 0{,}5$; 8) $d=a>0\wedge b>0$; 9) $w=d^2-4ac\geqslant 0\vee a\neq 0$; 10) $u=(a=b\vee c=d)\wedge a+b>2c)\wedge\neg\neg a>c$.

Здесь знаки $\neg$, $\wedge$, $\vee$ обозначают соответственно логические операции НЕ, И, ИЛИ.

Оператор STOP используется для завершения выполнения программы, когда вычисления закончены. Если снова продолжить вычисления, то выполнение программы начинается с первого оператора.

Оператор PAUSE также используется для прекращения вычисления. Однако если снова продолжить вычисление, то машина будет выполнять программу начиная с оператора, следующего за оператором PAUSE.

Оператор END используется для обозначения конца программы. По нему никаких действий машина не выполняет. Это самый последний оператор программы.

В программе сначала записываются невыполняемые операторы, а затем выполняемые. Исключение составляет невыполнимый оператор FORMAT, который может стоять в любом месте программы до оператора END.

Заканчивается программа оператором END.

Все переменные, используемые в программе, должны быть описаны либо по соглашению, либо операторами описания.

Исходные данные должны быть определены (заданы своими значениями) до их использования. Это достигается с помощью либо оператора присваивания, либо оператора ввода, либо оператора DATA, либо явного описания.

Операторы в программе должны располагаться в той последовательности, в которой они будут выполняться.

● **2.37.** Составить программу решения задачи 1.1.

Программа имеет вид

```
      READ (5,1) A, B, C
1     FORMAT (3F6.2)
```
} ввод исходных данных.

```
      P=(A+B+C)/2.
      T=2*SQRT(P*(P-A)*(P-B)*(P-C))
      HA=T/A
      HB=T/B
      HC=T/C
```
} вычисление высот треугольника.

```
      WRITE (6,2) HA, HB, HC
2     FORMAT (3E15.5)
      STOP
      END
```
} печать результатов.

2.38. Составить программу решения задачи 1.2. Ввод исходных данных осуществить по спецификации F3.1., печать результатов — по спецификации E12.5.

▲ **2.39.** Составить программу решения задачи 1.3. Значения x_i, y_i, m_i хранятся в массивах. Для ввода и вывода использовать спецификацию F4.1.

2.40. Составить программу решения задачи 1.4. Все переменные в программе — простые.

2.41. Составить программу решения задачи 1.5. Ввод и вывод осуществить по спецификации F3.1.

2.42. Составить программу решения задачи 1.6. Ввод осуществить по спецификации F5.2., а вывод на печать — по спецификации E12.5.

2.43. Составить программу решения задачи 1.7. Ввод осуществить по спецификации F5.3., а вывод на печать — по спецификации E12.4.

▲ **2.44.** Составить программу решения задачи 1.8. Ввод осуществить по спецификации F5.1., а вывод на печать — по спецификации F7.2.

2.4. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Для нарушения естественного порядка выполнения операторов используются управляющие операторы.

Оператор перехода GO TO n используется для организации безусловного перехода к оператору, имеющему метку n. В качестве метки используется целое число без знака, содержащее от одной до пяти цифр. Метка ставится перед оператором. Оператор, следующий за оператором GO TO n, обязательно должен иметь метку, так как в противном случае он не будет выполняться.

Условный оператор используется для изменения естественного порядка выполнения операторов после проверки некоторого условия. Различают условный арифметический и условный логический операторы.

Условный арифметический оператор имеет следующий вид:

IF(a) n_1, n_2, n_3,

где a — арифметическое выражение; n_1, n_2, n_3 — метки.

При $a<0$ осуществляется переход к оператору с меткой n_1, при $a=0$ — с меткой n_2 и при $a>0$ — с меткой n_3. Оператор, следующий за условным арифметическим оператором, должен иметь метку.

Условный логический оператор имеет следующий вид:

IF(b)s,

где b — логическое выражение, s — любой выполняемый оператор, кроме другого условного логического и оператора цикла.

Если логическое выражение b имеет значение .TRUE., то выполняется оператор s и далее оператор, следующий за условным, если оператор s не является управляющим и не осуществит перехода к другому месту программы. Если же b имеет значение .FALSE., то оператор s пропускается и выполняется оператор, следующий за условным.

● **2.45.** Составить программу решения задачи 1.9. Для реализации разветвления использовать условный арифметический опера-

тор, проверяющий условие $y=0$. Если $y=0$, то организовать переход к оператору с меткой 2, которую должен иметь оператор, осуществляющий вывод на печать $Y=0$. Если $y\neq 0$, то осуществить переход к оператору с меткой 3, который должен вычислять z. После печати z организовать переход к оператору STOP с помощью оператора GO TO 4. Программа решения этой задачи имеет вид:

```
   READ (5,1) X, N
1  FORMAT (F4.1, I2)
   Y=SIN (N*X)+0.5          — ввод исходных данных.
   IF(Y) 3, 2, 3— проверка условия.
3  Z=X**3/Y                 }
   WRITE (6,5) Z            } — первая ветвь программы.
5  FORMAT (E 12.4)          }
   GO TO 4          — переход к концу программы.
2  WRITE (6,6)              }
6  FORMAT (4H Y=0)          } — вторая ветвь программы.
4  STOP                     }
   END
```

В случае использования условного логического оператора программа будет иметь вид

```
   READ (5,1) X, N
1  FORMAT (F4.1, I2)
   Y=SIN (N*X)+0.5
   IF(Y.EQ.0)GO TO 2
   Z=X**3/Y
   WRITE (6,5) Z
5  FORMAT (E12.4)
   GO TO 4
2  WRITE (6,6)
6  FORMAT (4H Y=0)
4  STOP
   END
```

Если $y=0$, то осуществляется переход к оператору с меткой 2, в противном случае выполняется оператор, следующий за условным.

● **2.46.** Составить программу решения задачи 1.10, используя условный арифметический оператор.

Программа имеет вид

```
   READ (5,1) A, B, X
1  FORMAT (3F5.2)
   IF (X—A) 2, 2, 3
2  Z=SIN (X)                }
   GO TO 6                  } — первая ветвь программы.
3  IF (X—B) 5, 4, 4
4  Z=TAN (X)                } — вторая ветвь программы.
   GO TO 6                  }
5  Z=cos X                  } — третья ветвь программы.
6  WRITE (6,7) Z            }
7  FORMAT (F.8.4)
   STOP
   END
```

Если $x-a\leqslant 0$, то осуществляется переход к оператору с меткой 2, в противном случае — к оператору с меткой 3. Здесь проверяется

одно из оставшихся условий. Если $x-b \geqslant 0$, то организуется переход к оператору с меткой 4, в противном случае — к оператору с меткой 5. Затем обязательно осуществляется переход к оператору печати.

▲ **2.47.** Составить программу решения задачи 1.11, используя условный арифметический оператор.

2.48. Составить программу решения задачи 1.12, используя условный логический оператор.

▲ **2.49.** Составить программу решения задачи 1.13, используя условный арифметический оператор для вариантов а) и в) и условный логический оператор для варианта б).

▲ **2.50.** Составить программу решения задачи 1.14, используя условный логический оператор, если c — переменная логического типа.

▲ **2.51.** Составить программу решения задачи 1.15, используя условный арифметический оператор.

▲ **2.52.** Составить программу решения задачи 1.16, используя условный логический оператор.

2.53. Составить программу решения задачи 1.17, используя условный арифметический оператор.

▲ **2.54.** Составить программу решения задачи 1.18, используя а) условный арифметический оператор, б) условный логический оператор.

▲ **2.55.** Составить программу решения задачи 1.19. Целая часть от $x+0{,}5$ является ближайшим целым числом.

▲ **2.56.** Составить программу решения задачи 1.20. Число кратно трем, если целая часть частного от деления его на 3 после умножения на 3 будет равна исходному числу.

2.5. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Рассмотренных выше операторов вполне достаточно для организации цикла. Для организации цикла необходимо (см. гл. 1): задать перед циклом начальное значение параметра цикла (с помощью оператора присваивания); в цикле вычислить текущее значение параметра (с помощью оператора присваивания); проверить условия окончания цикла (с помощью логического выражения); управлять циклом (с помощью управляющего оператора).

● **2.57.** Составить программу решения задачи 1.21, алгоритм которой представлен на рис. 1.5, используя условный логический оператор.

Программа имеет вид

```
    READ (5,1) A
1   FORMAT (F4.1)
    X=9.                          — задает начальное значение параметра
                                    цикла.
2   Y=A**3/(A*A+X*X)
    WRITE (6,3) Y
3   FORMAT (E15.5)
```

```
X=X+0.1                  — вычисляет текущее значение параметра
                           цикла.
IF(X.LE.3) GO TO 2       — проверяет условие повторения цикла и
STOP                       осуществляет переход к его началу.
END
```

● 2.58. Составить программу решения задачи 1.22, алгоритм которой представлен на рис. 1.7, используя условный арифметический оператор.

Программа имеет вид

```
    READ (5,1) X, EPS
1   FORMAT (F4.1, E7.1)
    Y=1.
    N=1
2   Y=Y*X/N
    N=N+1
    WRITE (6,3) Y
3   FORMAT (F8.3)
    IF (Y—EPS) 4, 4, 2
4   STOP
    END
```

В задаче 2.58 цикл — итерационный. Для организации таких циклов наиболее удобен рассмотренный способ. В случае если количество повторений известно, программа получается более компактной и наглядной, если использовать для организаций таких повторений специальный оператор цикла.

Оператор цикла имеет вид

DO$ni=m_1, m_2, m_3$,

где n — метка последнего оператора, входящего в цикл, i — параметр цикла — простая переменная целого типа, начальное значение которой m_1, конечное значение m_2 и шаг изменения m_3 (m_1, m_2, m_3 — целые числа без знака или целочисленные переменные, значения которых должны быть определены до оператора цикла).

Если $m_3=1$, то оператор цикла имеет вид

DO$ni=m_1, m_2$

Последним оператором цикла не может быть управляющий оператор (за исключением условного логического, не содержащего управляющего оператора). В этом случае для обозначения конца цикла используется оператор CONTINUE, обязательно имеющий метку.

Входить в цикл можно только через оператор DO.

Выход из цикла можно осуществить двумя способами: естественным, когда цикл выполнен заданное количество раз (в этом случае после выхода из цикла параметр цикла не сохраняет своего значения); с помощью управляющего оператора, находящегося внутри цикла (в этом случае параметр цикла сохраняет свое значение).

Параметры, входящие в оператор DO, внутри цикла изменять нельзя.

● 2.59. Составить программу решения задач 1.23, используя оператор цикла.

Программа имеет вид

```
    DIMENSION A (40), X (40)
    READ (5,1) A, X
1   FORMAT (10 F8.3)
    DO 2 I=1,40
    Z=SQRT ((X(I)+A(I))/2.)      } — цикл
2   WRITE (6,3) Z
3   FORMAT (1X, F10.4)
    STOP
    END
```

Исходные массивы A и X описаны в операторе DIMENSION. Оператор DO организует цикл с параметром i, изменяющимся от 1 до 40 (шаг равен 1, поэтому не указывается). Внутри цикла повторяются операторы: присваивания, вычисляющий z, и печати.

2.60. Составить программу решения задачи 1.24, используя оператор цикла.

▲ **2.61.** Составить программу решения задачи 1.25, используя оператор цикла.

2.62. Составить программу решения задачи 1.26, используя оператор цикла.

▲ **2.63.** Составить программу решения задачи 1.27, используя условный логический оператор.

▲ **2.64.** Составить программу решения задачи 1.28, используя условный арифметический оператор.

2.65. Составить программу решения задачи 1.29, используя оператор цикла и условный логический оператор.

▲ **2.66.** Составить программу решения задачи 1.30, используя оператор цикла и условный арифметический оператор.

▲ **2.67.** Составить программу решения задачи 1.31, используя оператор цикла и условный логический оператор.

2.68. Составить программу решения задачи 1.32, используя оператор цикла.

▲ **2.69.** Составить программу решения задачи 1.33, используя условный логический оператор.

▲ **2.70.** Составить программу решения задачи 1.34, используя оператор цикла.

▲ **2.71.** Составить программу решения задачи 1.35, используя условный арифметический оператор.

2.72. Составить программу решения задачи 1.36, используя условный арифметический оператор.

▲ **2.73.** Составить программу решения задачи 1.37, используя условный логический оператор.

▲ **2.74.** Составить программу решения задачи 1.38, используя оператор цикла, считая $n \leqslant 30$.

2.75. Составить программу решения задачи 1.39, используя условный арифметический оператор. Цикл должен заканчиваться, когда $y \leqslant 0$.

▲ **2.76.** Составить программу решения задачи 1.40, используя оператор цикла. Ввести следующие обозначения: $c = \cos((n-1)x)$, $a = \cos x$, $b = \sqrt{1-\cos^2 x}$, $d = \sin((n-1)x)$.

▲ **2.77.** Составить программу решения задачи 1.41, используя условный логический оператор.

▲ **2.78.** Составить программу решения задачи 1.42, считая $n \leqslant 50$. Использовать оператор цикла и условный логический оператор (условие определения кратности трем см. в задаче 2.56).

Вычисления в цикле с несколькими одновременно изменяющимися параметрами

Оператор цикла языка ФОРТРАН позволяет задавать закон изменения только одного целочисленного параметра. Для задания закона изменения других параметров нужно использовать операторы присваивания, одни из которых перед циклом задают их начальные значения, а другие внутри цикла вычисляют их текущие значения. Если в цикле изменяется один параметр, который не является целочисленным, то он не может быть указан в операторе цикла. В этом случае необходимо вводить вспомогательный целочисленный параметр, который будет счетчиком количества повторений цикла.

● 2.79. Составить программу решения задачи 1.43.

Параметр i будем изменять с помощью оператора цикла, а переменную x — с помощью двух операторов присваивания:

```
      DIMENSION Y (20)
      READ (5,1) Y, A, H
1     FORMAT (10F5.2)
      X=A                          — задает начальное значение x.
      DO 2 I=1, 20
      Z=X*Y(I)/(X+Y(I))
      X=X+H                        — вычисляет текущее значение x.
2     WRITE (6, 3) Z
3     FORMAT (E12.4)
      STOP
      END
```

2.80. Составить программу решения задачи 1.44.

▲ 2.81. Составить программу решения задачи 1.45.

2.82. Составить программу решения задачи 1.46.

2.83. Составить программу решения задачи 1.47. Для организации цикла ввести искусственный целочисленный параметр.

▲ 2.84. Составить программу решения задачи 1.48. Начальное значение x и величину h задавать с помощью оператора ввода.

▲ 2.85. Составить программу решения задачи 1.49. Перед циклом задать начальные значения обоих параметров x и n.

Запоминание результатов

● 2.86. Составить программу решения задачи 1.50.

Для запоминания результатов необходимо выделить массив, который должен быть описан в операторе DIMENSION или операторах явного описания. В программе следует вычислять результат как переменную с индексом:

```
      DIMENSION X (50), Z (50) — описывает массив результатов Z.
      READ (5,1) X
1     FORMAT (10F8.3)
      DO 2 I=1,50
2     Z(I)=SQRT((X(I)**2+1.)/I)  — вычисляет результат как переменную с
                                   индексом.
      WRITE (6,3) Z
3     FORMAT (10E12.4)
      STOP
      END
```

▲ 2.87. Составить программу решения задачи 1.51, используя приемы запоминания результатов и организации цикла с несколькими одновременно изменяющимися параметрами.

2.88. Составить программу решения задачи 1.52.

▲ 2.89. Составить программу решения задачи 1.53.

2.90. Составить программу решения задачи 1.54, используя формулу $y_k = x_{41-k}$.

▲ 2.91. Составить программу решения задачи 1.55. Чтобы элементы записывались в массив Y подряд, необходимо изменять индекс переменной y_k на единицу всякий раз, когда осуществляется запись в этот массив.

2.92. Составить программу решения задачи 1.56 (см. пояснение к задаче 2.91).

▲ 2.93. Составить программу решения задачи 1.57. С помощью условного оператора проверять условие, что в массив Y записано 10 элементов, и выходить из цикла.

▲ 2.94. Составить программу решения задачи 1.58. Использовать приемы запоминания результатов и организации цикла с несколькими одновременно изменяющимися параметрами, полагая $]\,(x_m - x_0)/h[< 100$.

2.95. Составить программу решения задачи 1.59. Индексы элементов массивов A и B принимают разные значения.

▲ 2.96. Составить программу решения задачи 1.60. Индексы элементов массивов A, B и C принимают разные значения.

▲ 2.97. Составить программу решения задачи 1.61. Как только функция y станет отрицательной, вычисления прекратить.

▲ 2.98. Составить программу решения задачи 1.62. Индексы элементов массивов X, Y и Z принимают разные значения.

2.99. Составить программу решения задачи 1.63. Цикл организовать по n. Значения индексов элементов массивов A и B не совпадают с n.

2.100. Составить программу решения задачи 1.64. Элемент главной диагонали матрицы — переменная с двумя индексами, причем индексы одинаковые. Элемент одномерного массива — переменная с одним индексом.

Вычисление суммы и произведения

● 2.101. Составить программу решения задачи 1.65:

```
      DIMENSION X(20)
      READ (5,1) X
1     FORMAT (16F5.2)
      Z=0                          — задает начальное значение суммы.
      DO 2I=1,20
2     Z=Z+X(I)**2/I                — накапливает сумму.
      WRITE (6,3) Z
3     FORMAT (E12.3)
      STOP
      END
```

● 2.102. Составить программу решения задачи 1.66:

```
      DIMENSION X(100)
      READ (5,1) X
1     FORMAT (16F5.2)
      Z=1.                          — задает начальное значение произведения.
      DO 2 I=1,100
2     IF(X(I).GT.0) Z=Z*X(I)        — накапливает произведения положительных
                                      x_i.
      WRITE (6,3) Z
3     FORMAT (E15.5)
      STOP
      END
```

2.103. Составить программу решения задачи 1.67. Использовать прием накопления суммы и организации цикла с несколькими одновременно изменяющимися параметрами.

2.104. Составить программу решения задачи 1.68.

2.105. Составить программу решения задачи 1.69.

2.106. Составить программу решения задачи 1.70.

2.107. Составить программу решения задачи 1.71. Параметр цикла изменяется от 2 с шагом 2.

▲ **2.108.** Составить программу решения задачи 1.72. Начальное значение суммы равно 1, так как нет смысла вычислять первое слагаемое.

▲ **2.109.** Составить программу решения задачи 1.73. Вычислить сумму отрицательных элементов и их количество.

▲ **2.110.** Составить программу решения задачи 1.74. Вычислить сумму и количество элементов, удовлетворяющих условию $1 \leqslant a_i \leqslant 2$.

2.111. Составить программу решения задачи 1.75.

▲ **2.112.** Составить программу решения задачи 1.76, считая $n < 15$.

2.113. Составить программу решения задачи 1.77. Начальное значение суммы равно единице.

2.114. Составить программу решения задачи 1.78.

2.115. Составить программу решения задачи 1.79.

2.116. Составить программу решения задачи 1.80. Элементы главной диагонали имеют два индекса с одинаковыми значениями.

2.117. Составить программу решения задачи 1.81.

2.118. Составить программу решения задачи 1.82.

▲ **2.119.** Составить программу решения задачи 1.83. Вычислить произведение и количество элементов, больших a.

▲ **2.120.** Составить программу решения задачи 1.84. Вычислить произведение и количество положительных элементов. Параметр цикла изменяется от 2 с шагом 2.

▲ **2.121.** Составить программу решения задачи 1.85.

2.122. Составить программу решения задачи 1.86.

▲ **2.123.** Составить программу решения задачи 1.87.

▲ **2.124.** Составить программу решения задачи 1.88. Четное число делится без остатка на 2.

▲ **2.125.** Составить программу решения задачи 1.89. Целая часть числа $x_i + 0,5$ является ближайшим целым.

2.126. Составить программу решения задачи 1.90. Суммы вычислять в отдельных циклах.

Вычисление суммы членов бесконечного ряда

● **2.127.** Составить программу решения задачи 1.91. Цикл организовать с помощью условного оператора:

```
    READ (5,1) X, EPS
1   FORMAT (F.4.1, E6.1)
    Y=1.                          — задает начальное значение члена ряда.
    Z=1.                          — задает начальное значение суммы.
    N=1                           — задает начальное значение параметра
                                    цикла.
2   Y=Y*(—X*X)/((2*N—1)*2*N)      — вычисляет текущий член ряда.
    Z=Z+Y                         — накапливает сумму.
    N=N+1                         — изменяет параметр цикла.
    IF (Y.GT.EPS) GO TO 2         — проверяет условие повторения цикла.
    WRITE (6,3) Z
3   FORMAT (E12.4)
    STOP
    END
```

Точность, как правило, задается в виде константы с порядком, поэтому для ввода ε используется спецификация E.

2.128. Составить программу решения задачи 1.92.

2.129. Составить программу решения задачи 1.93. Для вычисления $m!$ организовать отдельный цикл.

2.130. Составить программу решения задачи 1.94.

2.131. Составить программу решения задачи 1.95.

2.132. Составить программу решения задачи 1.96. Для вычисления текущего значения члена ряда использовать рекуррентную формулу.

2.133. Составить программу решения задачи 1.97.

▲ **2.134.** Составить программу решения задачи 1.98, используя оператор цикла. Для вычисления слагаемого использовать рекуррентную формулу.

Вычисление полинома

● **2.135.** Составить программу решения задачи 1.99:

```
    DIMENSION A(9)
    READ (5,1) A, X
1   FORMAT (10F4.1)
    Y=A(1)                        — задает начальное значение полинома.
    DO 2 I=2,9
2   Y=Y*X+A(I)                    — вычисляет текущее значение полинома.
    WRITE (6,3) Y
3   FORMAT (E15.5)
    STOP
    END
```

Программа вычисления полинома любой степени n ($n \leqslant 30$) будет иметь вид

```
    DIMENSION A(30)
    READ (5,1) N, (A(I), I=1, N), X
1   FORMAT (I2/(20F4.1))
    Y=A(1)
    N1=N+1
```

```
      DO 2I=2, N1
2     Y=Y*X+A(I)
      WRITE(6,3) Y
3     FORMAT (E15.5)
      STOP
      END
```

2.136. Составить программу решения задачи 1.100.

2.137. Составить программу решения задачи 1.101.

▲ **2.138.** Составить программу решения задачи 1.102.

▲ **2.139.** Составить программу решения задачи 1.103.

▲ **2.140.** Составить программу решения задачи 1.104.

Нахождение наибольшего и наименьшего

● **2.141.** Составить программу решения задачи 1.105.

Особенность языка ФОРТРАН в том, что в операторе цикла можно указывать закон изменения только целочисленного параметра. В задаче параметр x — действительная переменная. Поэтому приходится вводить вспомогательный целочисленный параметр i, задающий последовательность повторений, количество которых определяется по формуле $n=](c-0)/h[+1$. Таким образом, в цикле одновременно будут изменяться два параметра: i и x.

```
      READ (5,1) A, B, C, H, OMEGA, FI
1     FORMAT (6F5.2)
      YMIN=1E19                    — задает начальное значение наименьшего.
      X=0.                         — задает начальное значение параметра x.
      N=C/H+1                      — вычисляет количество повторений цикла.
      DO 2 I=1, N
      Y=A*EXP(-B*X)*SIN(OMEGA*X+FI)
      X=X+H
2     IF(Y.LT.YMIN)YMIN=Y          — находит наименьшее.
      WRITE (6.3) YMIN
3     FORMAT (E12.4)
      STOP
      END
```

Условный логический оператор, в состав которого входит неуправляющий оператор, может быть последним оператором цикла. Если условие $y<y_{min}$ выполняется, то выполняется и оператор присваивания $y_{min}=y$, в противном случае он пропускается.

● **2.142.** Составить программу решения задачи 1.106:

```
      DIMENSION X (40)
      READ (5,1) X
1     FORMAT (10F8.2)
      XMAX=X(1)                    — задает начальное значение наибольшего.
      NMAX=1                       — задает начальное значение номера наибольшего.

      DO 2 I=2,40
      IF(X(I)-XMAX) 2, 2, 3        — проверяет условие x_i>x_max.
3     XMAX=X(I)                    }
      NMAX=I                       } — определяет наибольший элемент и его порядковый номер.
2     CONTINUE
      WRITE (6,4) XMAX, NMAX
4     FORMAT (F.9.2, 13)
```

```
      STOP
      END
```

Здесь для обозначения конца цикла используется оператор CONTINUE, по которому никаких действий не выполняется.

● **2.143.** Составить программу решения задачи 1.107.

Необходимо ввести целочисленный параметр i для записи оператора цикла, что приведет к дополнению схемы алгоритма (см. рис. 1.17) еще двумя блоками, один из которых перед циклом будет задавать начальное значение параметра x, а второй будет вычислять внутри цикла его текущее значение:

```
      READ (5,1) A, B, C, H
1     FORMAT (4F 4.1)
      IF(C) 2, 2, 3                    — проверяет знак c.
3     N=1
      GO TO 4
2     N=—1
4     YM=N*1E19                        — задает начальное значение ym
      X=0.                             — задает начальное значение параметра x.
      M=4/H+1                          — вычисляет количество повторения цикла.
      DO 5I=1, M
      V=ABS(A)*EXP(B*X+C*X*X)
      IF(N*Y—N*YM)6, 7, 7              — проверяет условие Ny<Nym, если оно не
6     YM=Y                               выполняется, то выход из цикла.
5     X=X+H                            — вычисляет текущее значение параметра x.
7     WRITE (6,8)N, YM
8     FORMAT(I3, E12.4)
      STOP
      END
```

2.144. Составить программу решения задачи 1.108.

2.145. Составить программу решения задачи 1.109.

2.146. Составить программу решения задачи 1.110.

▲ **2.147.** Составить программу решения задачи 1.111.

Для нахождения наибольшего и наименьшего значений функции перед циклом нужно задать начальное значение наибольшего и наименьшего. Поскольку функция достигает сначала максимума, в цикле надо сначала находить наибольшее, проверяя условие $y > y_{max}$. Как только условие $y > y_{max}$ не будет выполняться, надо переходить к отысканию наименьшего, т. е. переходить к проверке условия $y < y_{min}$. Если y окажется больше y_{min}, необходимо выходить из цикла.

2.148. Составить программу решения задачи 1.112 (см. пояснение к задаче 2.147).

▲ **2.149.** Составить программу решения задачи 1.113 (см. пояснение к задаче 1.107).

▲ **2.150.** Составить программу решения задачи 1.114.

2.151. Составить программу решения задачи 1.115. В цикле найти наибольший и наименьший элементы и их порядковые номера.

▲ **2.152.** Составить программу решения задачи 1.116.

2.153. Составить программу решения задачи 1.117. Функция достигает сначала максимума. Как только будет найден минимум, следует организовать выход из цикла (см. пояснение к задаче 2.147).

▲ 2.154. Составить программу решения задачи 1.118.
▲ 2.155. Составить программу решения задачи 1.119.

Уточнение корней уравнений

▲ 2.156. Составить программу решения задачи 1.120:

```
      X0=4.7                          — задание начального значения корня.
1     X1=ATAN (X0)+3.14159            — вычисление уточненного значения корня.
      IF(ABS(X1—X0).LE.1E—5)GO TO2 — проверка полученной точности вычислений.
      X0=X1                           — задание нового начального значения корня.
      GO TO 1                         — переход к началу цикла.
2     WRITE (6,3) X0, X1
3     FORMAT (2E15.7)
      STOP
      END
```

▲ 2.157. Составить программу решения задачи 1.121.
▲ 2.158. Составить программу решения задачи 1.122.
2.159. Составить программу решения задачи 1.123.
▲ 2.160. Составить программу решения задачи 1.124.

2.7. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ СО СТРУКТУРОЙ ВЛОЖЕННЫХ ЦИКЛОВ

В правильно организованном вложенном цикле операторы внутреннего цикла не могут выходить за пределы внешнего цикла, хотя внутренний и внешний циклы могут заканчиваться одним оператором.

● 2.161. Составить программу решения задачи 1.125:

```
        DIMENSION A (10,8)
        READ (5,1)A
      1 FORMAT (10F8.3)
  ——→   DO 2 I=1,10                  — внешний цикл.
  |     S=0.
  |  →  DO 3 J=1,8                   — внутренний цикл.
  |  |3 IF (A, (I, J).GT.0)S=S+A(I, J— конец внутреннего цикла.
  |__2  WRITE (6,4) S                — конец внешнего цикла.
      4 FORMAT (E15.5)
        STOP
        END
```

● 2.162. Составить программу решения задачи 1.126.

Поскольку в операторе цикла нельзя записывать выражения в отличие от схемы алгоритма, представленного на рис. 1.20, перед внутренним циклом необходимо вычислить $k1=k+1$:

```
      DIMENSION X(100)
      READ (5,1)X
1     FORMAT (20F4.1)
```

```
      DO2K=1,99
      XMIN=X(K)
      N=K
      K1=K+1
      DO3 J=K1,100
      IF(X(J)—XMIN) 4, 3, 3
    4 XMIN=X(J)
      N=J
    3 CONTINUE
      X(N)=X(K)
    2 X(K)=XMIN

      WRITE (6,5)X
    5 FORMAT (20F6.1)
      STOP
      END
```

2.163. Составить программу решения задачи 1.127. При вычислении одной суммы оператор, задающий начальное значение суммы равное 0, должен выполняться один раз, т. е. стоять перед внешним циклом. По этой же причине оператор печати тоже должен стоять за внешним циклом.

▲ **2.164.** Составить программу решения задачи 1.128. На печать вывести сразу весь массив результатов.

▲ **2.165.** Составить программу решения задачи 1.129. Индекс номера строки i изменяется от 1 до 19, так как в 20-й строке нет элемента, лежащего над главной диагональю, а индекс номера столбца j от $j+1$ до 20.

2.166. Составить программу решения задачи 1.130. В цикле необходимо находить наибольшие элементы массива начиная с первого, второго элемента и т. д.

2.167. Составить программу решения задачи 1.131. Результат решения отпечатать массивом.

2.168. Составить программу решения задачи 1.132. На печать выводить каждое вновь вычисленное среднего арифметического.

▲ **2.169.** Составить программу решения задачи 1.133. Во внутреннем цикле находить сумму элементов строки, а во внешнем — наименьшую из этих сумм и номер ее строки.

2.170. Составить программу решения задачи 1.134.

▲ **2.171.** Составить программу решения задачи 1.135. Результат решения отпечатать массивом.

2.172. Составить программу решения задачи 1.136.

▲ **2.173.** Составить программу решения задачи 1.137. Результат решения отпечатать массивом.

2.174. Составить программу решения задачи 1.138. Во внутреннем цикле находить наибольший элемент массива и его порядковый номер. Во внешнем цикле записать этот элемент в очередную ячейку массива Y, а на место наибольшего элемента в массив X записывать число порядка -10^{10}, чтобы этот элемент снова не был найден как наибольший.

2.175. Составить программу решения задачи 1.139.

▲ **2.176.** Составить программу решения задачи 1.140. Организовать сложный цикл для нахождения наименьшего элемента матри-

цы и номера строки и столбца, в которых он находится. Затем организовать два простых цикла для записи нулей в эту строку и столбец. Матрицу распечатать в общепринятом виде.

▲ **2.177.** Составить программу решения задачи 1.141.

2.178. Составить программу решения задачи 1.142. Во внешнем цикле три раза повторить цикл для нахождения наибольшего элемента массива. Вместо наибольшего элемента в исходный массив записать число порядка -10^{10}.

▲ **2.179.** Составить программу решения задачи 1.143. На печать бывести результирующую матрицу в общепринятом виде (n, $l \leqslant 10$, $m \leqslant 20$).

2.180. Составить программу решения задачи 1.144. Организовать внешний цикл, выполняемый три раза, в котором будут осуществляться действия, начиная с ввода массива и кончая печатью результатов.

▲ **2.181.** Составить программу решения задачи 1.145.

2.182. Составить программу решения задачи 1.146.

● **2.183.** Составить программу решения задачи 1.147.

Программа имеет вид

```
      READ (5,1)A
    1 FORMAT (F.4.2)
      YMIN=1.E19
      H=0.2
      X=0.2
    6 DO 2 I=1,5 0
      Y=A*X—ALOG(X)
      IF(Y—YMIN) 3, 4, 4
    3 YMIN=Y
      XMIN=X
    2 X=X+H
    4 IF(H.LE.0.01)GO TO 5
      X=XMIN—0.2
      H=0.01
      GO TO 6
    5 WRITE (6,7) XMIN
    7 FORMAT (E15.7)
      STOP
      END
```

Здесь внешний цикл выполняется два раза. Для его организации использованы условный логический оператор и оператор GO TO 6. Выход из этого цикла при условии, что $h=0{,}01$.

2.184. Составить программу решения задачи 1.148. При решении использовать методику решения задачи 2.183.

▲ **2.185.** Составить программу решения задачи 1.149. Матрицу отпечатать в общепринятом виде.

▲ **2.186.** Составить программу решения задачи 1.150. При решении использовать методику решения задачи 2.183.

▲ **2.187.** Составить программу решения задачи 1.151. Во внутреннем цикле надо найти сначала максимум, а потом минимум функции. Если минимум пройден, то осуществить выход из внутреннего цикла, запись максимума и минимума в массив Y и повторение внеш-

него цикла. Внутренний и внешний циклы организовать с помощью условного оператора. Условие окончания внешнего цикла $x>5$.

2.188. Составить программу решения задачи 1.152.

2.8. ОРГАНИЗАЦИЯ ПОДПРОГРАММ

В процессе решения многих задач необходимо многократно производить одни и те же вычисления при различных значениях параметров. В этом случае для уменьшения размера программы целесообразно выделить эти вычисления в подпрограмму, где описывается процедура вычисления некоторой функции при формальных параметрах. В основной программе в нужных местах осуществляется обращение к подпрограмме при заданных значениях фактических параметров. В языке ФОРТРАН используются подпрограммы: оператор-функция, подпрограмма-функция, подпрограмма общего вида.

Оператор-функция

Оператор-функция применяется для записи подпрограммы, состоящей из одного оператора, и записывается так: $F(a_1, a_2, ..., a_n)=E$, где F — имя подпрограммы, a_i — формальные параметры, E — выражение. Формальные параметры могут быть только простыми переменными. Тип функции определяется по соглашению или операторами явного описания или оператором IMPLICIT. Обращение к оператору-функции осуществляется с помощью указателя функции $F(b_1, b_2, ..., b_n)$, которая, как любая другая функция, входит в состав выражения (F — имя функции, b_i — фактические параметры (константы, простые или индексные переменные, выражения)).

Формальные и фактические параметры должны быть согласованы по количеству, порядку следования и типу.

Располагается оператор-функция в основной программе перед первым выполняемым оператором.

● 2.189. Составить программу вычисления функции $z=\frac{\log_2 x+\log_b y}{2\log_{(b+2)}(x+y)}$. Для вычисления логарифма использовать оператор-функцию. Любой логарифм определяется через натуральный логарифм $\log_a z=\ln z/\ln a$. Эту формулу будем использовать в операторе-функции для вычисления логарифма по любому основанию. Для обращения к оператору-функции необходимо задать два параметра: основание логарифма и число, от которого он вычисляется.

Присвоив этой подпрограмме имя ALG, получим

```
ALG(A, Z)=ALOG(Z)/ALOG(A)
```

В программе с помощью указателя функции нужно три раза обратиться к оператору-функции для вычисления трех логарифмов, используя следующие фактические параметры: ALG (2, X); ALG (B, Y); ALG (B+2, X+Y).

Программа имеет вид

```
   ALG(A, Z)=ALOG(Z)/ALOG(A)  — оператор-функция.
   READ (5,1) B, X, Y
1  FORMAT (3F5.1)
   Z=(ALG(2, X)+ALG(B, Y))/ALG(B+2, X+Y)/2.
   WRITE (6,2) Z
2  FORMAT (E15.7)
   STOP
   END
```

Подпрограмма-функция

Подпрограмма-функция служит для организации подпрограмм, допускающих любое количество операторов, но вычисляющих только одно значение, которое присваивается имени подпрограммы. Структура подпрограммы-функции следующая:

```
t FUNCTION F*s(a1, a2, ..., an)
<операторы подпрограммы>
F=...
<операторы подпрограммы>
RETURN
END
```

где t — указатель типа функции (REAL, INTEGER, LOGICAL, COMPLEX). Если этот указатель отсутствует, то тип функции определяется по соглашению; s — указатель длины, если этого указателя нет, то длина стандартная; RETURN — оператор возврата в основную программу; F — имя подпрограммы; a_i — формальные параметры, в качестве которых могут быть использованы простые переменные, имена массивов или подпрограмм. В подпрограмме должен быть хотя бы один оператор присваивания, в левой части которого стоит имя подпрограммы. Если среди формальных параметров есть имя массива, то этот массив в подпрограмме должен быть описан.

Подпрограмма-функция оформляется в виде отдельной программной единицы и располагается после основной программы (после оператора END) или после оператора END предыдущей программы.

Для обращения к подпрограмме-функции используется указатель функции $F(b_1, b_2, ..., b_n)$, где F — имя подпрограммы; b_i — фактические параметры (константы, простые или индексные переменные, массивы, выражения, имена других подпрограмм). Фактические параметры должны согласовываться с формальными по количеству, порядку следования и типу.

● **2.190.** Составить программу вычисления функции $y=ax^2+bx+c$, где $a=\sum_{i=1}^{n} t_i$, $b=\sum_{i=n+1}^{100} t_i$, $c=\sum_{i=1}^{20} q_i$;

t_i и q_i — элементы массивов.

Для вычисления функции использовать подпрограмму-функцию.

Присвоим подпрограмме имя SUM. Для того чтобы можно было вычислить по этой подпрограмме любую из приведенных выше сумм, необходимо среди формальных параметров указать имя массива и пределы суммирования. В общем случае подпрограмма должна вычислять $s=\sum_{i=k}^{l} z_i$.

В основной программе необходимо трижды обратиться к этой подпрограмме для вычисления a, b, c при соответствующих значениях фактических параметров:

```
      DIMENSION T (100), Q (20)
      READ (5,1) N, T, Q, X
1     FORMAT (I2/(20F4.1))
      Y=SUM (T, 1, N)*X*X+SUM(T, N+1, 100)*X+SUM(Q, 1, 20)
      WRITE (6,2)Y
2     FORMAT (E15.7)
      STOP
      END
      FUNCTION SUM (Z, K, L)
      DIMENSION Z(L)
      S=0.
```

```
      DO 1 I=K, L
1     S=S+Z(I)
      SUM=S
      RETURN
      END
```

С помощью указателя функции осуществляется обращение к подпрограмме для вычисления суммы элементов массива T от 1 до N (SUM (T, 1, N), суммы элементов массива T от N+1 до 100 (SUM(T, N+1, 100)) и суммы элементов массива Q от 1 до 20 (SUM (Q, 1, 20)).

В подпрограмме используется массив Z, который описан в операторе DIMENSION.

Оператор SUM=S присваивает значение суммы имени подпрограммы.

Оператор RETURN осуществляет возврат в основную программу.

Подпрограмма общего вида

Подпрограмма общего вида используется для получения в подпрограмме нескольких выходных результатов.

Структура подпрограммы общего вида следующая:

SUBROUTINE $F(a_1, a_2, ..., a_n)$
<операторы подпрограммы>
RETURN
END

Тип подпрограммы не специфируется, так как разные результаты ее выполнения могут иметь разный тип.

Для обращения к подпрограмме общего вида используется специальный оператор

CALL $F(b_1, b_2, ..., b_n)$,

где F — имя подпрограммы, b_i — фактические параметры.

Правила записи формальных и фактических параметров здесь такие же, как и для подпрограммы-функции.

● **2.191.** Составить программу вычисления функции

$$z = x^{k_1} y^{k_2} / (s_1 + s_2),$$

где s_1 и k_1 — сумма и количество положительных элементов массива $(a_1, a_2, ..., a_{70})$; s_2 и k_2 — сумма и количество элементов массива $(b_1, b_2, ..., b_{40})$.

Для вычисления суммы и количества элементов массива использовать подпрограмму общего вида.

Здесь два выходных параметра (результата): сумма и количество положительных элементов массива (эти параметры разного типа).

В подпрограмме будем вычислять сумму s положительных элементов массива X и их количество n. Количество элементов массива обозначим m:

```
      DIMENSION A (70), B (40)
      READ (5,1) A, B, X, Y
1     FORMAT (20F4.1)
      CALL F (A, 70, S1, K1)
      CALL F (B, 40, S2, K2)
```

```
      Z=X**K1*Y**K2/(S1+S2)
      WRITE (6,2)Z
2     FORMAT (E15.5)
      STOP
      END
      SUBROUTINE F (X, M, S, N)
      DIMENSION X (M)
      N=0
      S=0.
      DO 1 I=1, M
       IF (X(I).LE.0.) GO TO 1
      S=S+X(I)
      N=N+1
1     CONTINUE
      RETURN
      END
```

При первом обращении к подпрограмме по оператору CALL вместо массива X будет подставлен массив A и будут вычислены сумма и количество положительных элементов, которые будут присвоены переменным S1 и K1. При втором обращении к подпрограмме будут определены S2 и K2 для массива B.

По оператору RETURN осуществляется возврат в основную программу к оператору, следующему за оператором CALL, вызывающим подпрограмму.

Оператор EXTERNAL

В список фактических параметров подпрограммы могут входить имена внешних функций и подпрограмм, которые обязательно должны быть перечислены в операторе EXTERNAL.

Общий вид оператора следующий:

EXTERNAL $a_1, a_2, \ldots, a_n$

где a_i — имена внешних функций или подпрограмм, которые являются фактическими параметрами. Оператор EXTERNAL помещается перед первым выполняемым оператором основной программы.

● **2.192.** Составить программу для вычисления функции

$$z=\left(\sum_{i=1}^{20}\cos a_i+\sum_{i=1}^{30}\sin^2 b_i\right)\Big/\sqrt{\ln\sum_{i=1}^{40}|c_i|}.$$

Для вычисления суммы использовать подпрограмму общего вида.

В подпрограмме необходимо один раз вычислить сумму косинусов, второй раз — сумму синусов, третий раз — сумму модулей элементов некоторых массивов. В подпрограмме следует описать вычисление суммы некоторой функции F от аргумента x_i, являющегося элементом массива X, т. е. в подпрограмме будет вычисляться

$$R=\sum_{i=1}^{n}F(x_i).$$

При первом обращении к подпрограмме вместо F надо подставить COS, при втором — SIN и при третьем — ABS, т. е. имя функции надо передать в подпрограмму как фактический параметр.

Имена этих функций должны быть перечислены в операторе EXTERNAL.

С учетом сказанного программа решения задачи имеет вид

```
      DIMENSION A(20), B(30), C(40)
      EXTERNAL COS, SIN, ABS
      READ (5,1) A, B, C
1     FORMAT (10 F8.3)
      CALL SUM(COS, A, 20, R1)
      CALL SUM(SIN, B, 30, R2)
      CALL SUM (ABS, C, 40, R3)
      Z=(R1+R2)/ALOG(R3)
      WRITE (6,2)Z
2     FORMAT (E15.5)
      STOP
      END
      SUBROUTINE SUM (F, X, N, R)
      DIMENSION X(N)
      R=0.
      DO 1 I=1, N
1     R=R+F(X(I))
      RETURN
      END
```

При первом обращении к подпрограмме вместо F будет подставлено COS, вместо X подставлено A, вместо N подставлено 20 и будет получен результат R1. После других обращений аналогично получим R2 и R3. В исходном выражении есть еще одна функция ln, которая в операторе EXTERNAL не описана, так как не является фактическим параметром.

Дополнительные входы в подпрограмму

В рассмотренных примерах при обращении к подпрограмме выполнялись все операторы начиная с первого. Иногда появляется необходимость выполнять только часть подпрограммы. Например, в подпрограмме вычисляется $z=f_1+f_2$. При обращении к ней будут выполняться операторы, вычисляющие f_1 и f_2. Если необходимо при обращении к подпрограмме исключить вычисление f_1 и сразу приступить к вычислению f_2, то целесообразно иметь дополнительный вход, определяющий, с какого места должна выполняться подпрограмма. Для обозначения дополнительного входа используется невыполняемый оператор ENTRY, который имеет структуру

ENTRY $F1$ $(c_1, c_2, ..., c_n)$,

где $F1$ — имя части подпрограммы, которое должно отличаться от имени всей подпрограммы; c_i — формальные параметры, используемые в части подпрограммы.

Обращение к части подпрограммы осуществляется по ее имени $F1$ с указанием фактических параметров, которые согласуются с c_i. Дополнительные входы могут иметь и подпрограмма-функция, и подпрограмма общего вида.

● **2.193.** Составить программу для вычисления функции

$$z=\begin{cases} c+m^{-1}\sum\limits_{i=1}^{m} x_i, & \text{если } d\leqslant 0;\\ \sqrt[n]{\prod\limits_{i=1}^{n} y_i}+m^{-1}\sum\limits_{i=1}^{m} x_i, & \text{если } d>0. \end{cases}$$

Вычисление значений выражения осуществить в подпрограмме. Все $y_i > 0$.

Если в подпрограмме вычислять сначала среднее геометрическое из y_i, а затем среднее арифметическое из x_i и получать их сумму, то эта подпрограмма реализует вторую ветвь вычислительного процесса. Если организовать дополнительный вход перед вычислением среднего арифметического из x_i, то эта часть подпрограммы может быть использована для реализации первой ветви.

Программа имеет вид

```
    DIMENSION X(100), Y(100)
    READ (5.1) N, M, (X(I), I=1, M), (Y(I), I=1, N), C, D
1   FORMAT (2I3/(20F4.1))
    IF(D) 2, 2, 3
3   CALL SR (X, Y, N, M, Z)
    GO TO 4
2   CALL SRAR (X, M, C, Z)
4   WRITE (6,5) Z
5   FORMAT (E14.6)
    STOP
    END
    SUBROUTINE SR (X, Y, N, M, Z)
    DIMENSION X(M), Y(N)
    P=1.
    DO 1 I=1, N
1   P=P*Y(I)
    P=P**(1./N)
    ENTRY SRAR (X, M, P, Z)
    S=0.
    DO 2 I=1, M
2   S=S+X(I)
    S=S/M
    Z=P+S
    RETURN
    END
```

Если $d > 0$, вычисляется сначала P — среднее геометрическое элементов массива Y, затем S — среднее арифметическое массива X и, наконец, Z, равное их сумме. Если $d \leqslant 0$, то вычисляется сразу S — среднее арифметическое, вместо значения P подставляется значение C и вычисляется Z=C+S.

Дополнительные выходы из подпрограммы

Во всех рассмотренных подпрограммах возврат в основную программу осуществлялся с помощью оператора RETURN к месту, откуда было осуществлено обращение к подпрограмме. Подпрограмма общего вида позволяет возвращаться в любое место основной программы. Для этого используется оператор RETURN$_i$, где i — целочисленная переменная или целая константа без знака.

Для обозначения места в основной программе, куда следует вернуться после выполнения подпрограммы, среди фактических параметров надо указывать метки операторов, к которым будет происходить возврат. Эти метки в списке фактических параметров помещаются после символа & (амперсанд). На соответствующем месте в списке формальных параметров записывается *. Если в списке фактических параметров n меток, то i принимает значения 1, 2, ..., n. Выход по оператору RETURN k будет происходить к метке, стоящей на k-м месте в списке меток фактических параметров. Например,

```
CALL F(X, Y, N, Z, &11, &7, &16, &22).
```

По оператору RETURN 3 будет осуществлен возврат к оператору с меткой 16 основной программы.

● **2.194.** Составить программу вычисления значений функций $u=\sin t$, если $t>0$; $v=\cos\left(t+\frac{\pi}{4}\right)$, если $t<0$; $w=\sqrt{\left|\frac{x_1+x_{100}}{2}\right|}$, если $t=0$; где $t=\frac{1}{100}\sum_{i=1}^{100}x_i$. Определение аргумента этих функций выполнить в подпрограмме.

В подпрограмме накопим сумму и в зависимости от знака суммы определим аргумент Y, равный или t, или $t+\pi/4$, или $(x_1+x_{100})/2$. После этого следует выйти из подпрограммы и в зависимости от значения аргумента, возвратясь в основную программу, вычислить функцию или u, или v, или w. Следовательно, при различных значениях аргумента надо возвращаться в разные места основной программы. К вычислению одной функции можно вернуться по оператору RETURN, к вычислению других функций — используя дополнительные выходы.

С учетом сказанного программа имеет вид

```
      DIMENSION X(100)
      READ (5,1) X
1     FORMAT (10F8.3)
      CALL ARG (X, 100, 3.14159/4, Y, &4, &5)
      U=SIN(Y)
      WRITE (3,2) U
      GO TO 3
4     V=COS(Y)
      WRITE (3,2) V
      GO TO 3
5     W=SQRT (ABS (Y))
      WRITE (6,2) W
2     FORMAT (E12.4)
3     STOP
      END
```

```
      SUBROUTINE ARG (A, N, C, Z, *, *)
      DIMENSION A(N)
      Z=0.
        DO 1 I=1, N
1     Z=Z+A(I)
      Z=Z/N
      IF(Z) 2, 3, 4
2     Z=Z+C
      RETURN 1
3     Z=(A(1)+A(N)/2.
      RETURN 2
4     RETURN
      END
```

Если $z>0$, то осуществляется переход к оператору с меткой 4 и по оператору RETURN — выход в основную программу к оператору, следующему за оператором CALL, т. е. к вычислению U. При $z<0$ организуется переход к оператору с меткой 2, который к z прибавляет c, равное $\pi/4$, и по оператору RETURN1 передает управление в основную программу к оператору с меткой 4, указанной на первом месте в списке меток, т. е. к вычислению v.

Если $z=0$, то вычисляется $z=(x_1+x_{100})/2$ и происходит возврат в основную программу к оператору с меткой 5, указанной на втором месте в списке меток, т. е. происходит возврат к вычислению w.

Общие области памяти

Ограниченный объем оперативной памяти делает необходимым выделение одного и того же места для хранения нескольких переменных или массивов. Это достигается с помощью операторов EQUIVALENCE и COMMON.

Оператор EQUIVALENCE используется для выделения общей памяти двум (или более) переменным одной программной единицы.

Общий вид оператора следующий:

EQUIVALENCE $(a_1, a_2, ..., a_n), (b_1, b_2, ..., b_m), ...$

где a_i, b_i — простые или индексные переменные.

Всем переменным одной группы, заключенной в скобки, отводится для хранения один и тот же участок памяти. Переменные одной группы могут быть различного типа, но обязательно одинаковой длины.

Эквивалентность двух элементов разных массивов приводит к выделению общей памяти и для других элементов этих массивов. Например,

```
DIMENSION A(8), B(3,3)
EQUIVALENCE (A(1), B(1,1))
```

Оператор выделит общую память для хранения элементов A(1) и B(1,1), A(2) и B(2,1) A(3) и B(3,1) ... A(8) и B(2,3) массивов. Последний элемент B(3,3) массива будет храниться в отдельной ячейке, так как эквивалентного ему элемента в массиве A нет.

Оператор COMMON выделяет общую область памяти для хранения переменных различных программных единиц, например основной программы и подпрограммы. При этом оператор COMMON записывается в каждой программной единице.

Общий вид оператора следующий:

COMMON $c_1, c_2, ..., c_n$,

где c_i — имя переменной или массива.

Массив в операторе может указываться с размерностью, поэтому надобность описывать этот массив в других операторах отпадает. Тип согласуемых переменных может быть разным, но длина обязательно одинаковая.

Согласуемые массивы должны иметь одинаковую размерность.

Операторами COMMON A, B, C (20) и COMMON X, Y, Z (20) выделяется общая область для хранения переменных *a* и *x*, *b* и *y*, а также массивов C и Z.

Переменные, указанные в операторах COMMON и EQUIVALENCE, не могут быть формальными параметрами. Это часто используется для передачи параметров в подпрограмму, которая может стать подпрограммой без параметров.

● **2.195.** Составить программу решения задачи 1.144, используя для вычисления математического ожидания и дисперсии подпрограмму без параметров. В целях экономии памяти массивы A, B, C и формальный массив X хранить в одной области памяти.

В основной программе с помощью оператора EQUIVALENCE выделяется общая память для массивов A, B, C, а с помощью операторов COMMON — общая память для массивов A и X. Чтобы подпрограмма была без параметров, в операторах COMMON указываются все остальные фактические и формальные параметры. С учетом сказанного программа имеет вид

```
      REAL MX
      COMMON M, MX, DX, A(100)
       DIMENSION B(100), C(100)
      EQUIVALENCE (A(1), B(1), C(1))
      DO 1 I=1,3
      READ (5,2) M, (A(I), I=1, M)
2     FORMAT (I3/(10F8.3))
      CALL STAT
1     WRITE (6,3) MX, DX
3     FORMAT (2E15.6)
      STOP
      END
      SUBROUTINE STAT
```

```
      COMMON N, Y, Z, X(100)
      Y=0.
      DO 1 I=1, N
1     Y=Y+X(I)
      Y=Y/N
      Z=0.
      DO 2 I=1, N
2     Z=Z+(X(I)—Y)**2
      Z=Z/N
      RETURN
      END
```

Программа может быть использована для вычисления m_x и D_x трех массивов размерностью M, не более 100 элементов каждый. В основной программе для обозначения массивов можно использовать имя или A, или B, или C.

Поскольку массивы A, B, C хранятся в одной области памяти, вводиться они должны поочередно. Все параметры в подпрограмме задаются с помощью оператора COMMON.

2.196. Составить программу вычисления значения функции

$$s=\sqrt{x^2+y^2+\sin^2 xy}+\sqrt{x^2+z^2+\sin^2 xz}+\sqrt{y^2+z^2+\sin^2 yz},$$

используя оператор-функцию.

2.197. Составить программу вычисления функций

$$a=\frac{\sqrt{sz^3+qz^2+tz+t}}{1+e^{sz^3}+z-1},\quad b=\frac{t^2\sin\alpha+t\cos\beta+3.5}{r^2+2r-t}.$$

Для определения многочленов использовать оператор-функцию $y(a, b, c, d, x)=ax^5+bx^2+cx+d$.

▲ **2.198.** Составить программу вычисления функции

$$z=\frac{e^{a^2-1}\sqrt{\cos b}}{1-e^{b^2}}+\frac{\sqrt[3]{\cos(a+b)}+0.5}{2\sqrt{e^{c^2+1}}},$$

используя два оператора-функции.

▲ **2.199.** Составить программу вычисления приближенного значения функции $y=\sqrt[3]{x}$ по итерационной формуле $y_{n+1}=y_n+(1/3)\times(x/y_n^2-y_n)$ с точностью $\varepsilon=10^{-5}$, используя оператор-функцию. Начальное значение функции вычислить по формуле

$$y_0=\begin{cases} x/3, & \text{если } x\geqslant 1;\\ 3x, & \text{если } x<1.\end{cases}$$

▲ **2.200.** Составить программу решения задачи 1.79, используя подпрограмму-функцию:

$$n!=\begin{cases} 0, & \text{если } n<0;\\ 1, & \text{если } n=0;\\ n!, & \text{если } n>0.\end{cases}$$

2.201. Составить программу вычисления среднего арифметического положительных элементов массивов X(60), Y(75), Z(80), используя подпрограмму-функцию. В массивах есть положительные элементы. Массивы X, Y, Z хранить в одной области памяти.

2.202. Составить программу вычисления среднего геометрического положительных элементов каждого столбца матрицы $A(10\times20)$, используя подпрограмму-функцию. Фактическую и формальную матрицы хранить в одной области памяти. В каждом столбце есть положительные элементы.

2.203. Составить программу вычисления $z=\dfrac{e^{|x_{max}|}-e^{|y_{max}|}}{\sqrt{|x_{max}x_{min}|}}\times$ $\times\sqrt{|y_{max}y_{min}|}$, где x_{max} и x_{min} — наибольший и наименьший элементы массива X (100); y_{max} и y_{min} — наибольший и наименьший элементы массива Y (120). Нахождение наибольшего и наименьшего осуществить в одной подпрограмме-функции, причем результатом выполнения подпрограммы должно быть $R=A_{min}\cdot A_{max}$. Если надо найти только A_{max}, то $R1=1\cdot A_{max}$, для чего использовать дополнительный вход.

▲ **2.204.** Составить программу вычисления значения функции

$$z=\left(\sum_{i=1}^{40}\frac{\sqrt{a_i}}{a^2_{min\,i}}+\sqrt{\sum_{i=1}^{60}\frac{|b_i|}{b^2_{min\,i}}}\right)\Bigg/2,$$

где $a_{min\,i}$ и $b_{min\,i}$ — минимальные значения первых i элементов массивов A(40) и B(60), используя подпрограмму-функцию.

Для вычисления сумм использовать подпрограмму SUM, а для вычисления квадрата наименьшего — подпрограмму MIN. Все $a_i>0$.

2.205. Составить программу для вычисления суммы положительных элементов матрицы $A(n\times m)$ и их количества ($n\leqslant30$, $m\leqslant30$), используя подпрограмму общего вида.

▲ **2.206.** Составить программу вычисления значения функции $u=e^{x_1+y_1}-e^{x_2+y_2}$, где x_1, x_2 — корни уравнения $ax^2+bx-1{,}5=0$; y_1, y_2 — корни уравнения $2y^2-y+c=0$. Корни находить в подпрограмме. Если корни мнимые, то считать их равными нулю.

2.207. Составить программу вычисления и запоминания сумм элементов каждой строки матрицы $A(n\times m)$, используя подпрограмму общего вида ($n\leqslant30$, $m\leqslant30$).

▲ **2.208.** Составить программу определения наибольшего общего делителя двух целых положительных чисел x и y путем вычисления $a_1=\max(x, y)$; $a_2=\min(x, y)$; $a_3=\max(a_1-a_2, a_2)-\min(a_1-a_2, a_2)$, ...; $a_i=\max(a_{i-2}-a_{i-1}, a_{i-1})-\min(a_{i-2}-a_{i-1}, a_{i-1})$. Если $a_n=a_{n-2}$, то a_n—наибольший общий делитель. Для нахождения max и min использовать подпрограмму общего вида. Здесь в каждый момент имеются три значения: a_i, a_{i-1}, a_{i-2}. Обозначим их a_3, a_2, a_1. Тогда $a_3=\max(a_1-a_2, a_2)-\min(a_1-a_2, a_2)$. Начальные значения $a_1=\max(x, y)$ и $a_2=\min(x, y)$.

▲ **2.209.** Составить программу вычисления $z=x_1+x_2+x_3$, где

$$x_1=\left(\sum_{i=1}^{10}a_{2i+1}\right)\Bigg/21!,\quad x_2=\left(\sum_{i=1}^{20}b_{2i}\right)\Bigg/40!,\quad x_3=\left(\sum_{i=1}^{40}c_{i+1}\right)\Bigg/41!$$

(a_i, b_i и c_i — элементы массивов).

Вычисление x_i осуществить в подпрограмме общего вида. Для определения факториала использовать подпрограмму-функцию.

▲ **2.210.** Составить программу вычисления математического ожидания m_x и дисперсии D_x для моментов появления случайных событий $t_1, t_2, \ldots, t_{200}$ и интервалов между ними. Для вычисления m_x и D_x использовать подпрограмму без параметров. Перед вычислением m_x и D_x для интервалов необходимо упорядочить по возрастанию элементы массива T, вычислить и записать в массив T1 интервалы между смежными t_i ($T1_i = t_{i+1} - t_i$).

▲ **2.211.** Составить программу вычисления значений функций

$$t = \frac{y a_{\max}}{b_{\max} + b_{\min}}, \text{ если } x = \sum_{i=1}^{40} a_i < y = \sum_{i=1}^{50} b_i;$$

$$r = \frac{x a_{\min}}{b_{\max} + b_{\min}}, \text{ если } x \geqslant y,$$

где $a_{\max}$, $b_{\max}$, $a_{\min}$, $b_{\min}$ — соответственно наибольший и наименьший элементы массивов A и B. Вычисление сумм и проверку условия $x<y$ осуществить в одной подпрограмме с именем SUM. В другой подпрограмме предусмотреть нахождение наибольшего, наименьшего или их суммы, для чего использовать дополнительный вход. Имя этой подпрограммы EKSTR. В качестве одного из выходных параметров подпрограммы вычисления сумм предусмотреть параметр k, который принимает значение 1, если $x<y$, и значение 0 в противном случае. В подпрограмме SUM предусмотреть дополнительный выход.

В основной программе сначала обратиться к подпрограмме EKSTR для вычисления $b_{\max}+b_{\min}$, затем к подпрограмме SUM. Выход из этой подпрограммы будет либо к оператору, следующему за оператором CALL SUM, если $x<y$, либо к оператору, метка которого указана среди фактических параметров подпрограммы SUM. В первом случае необходимо обратиться к подпрограмме EKSTR для вычисления $a_{\max}$, во втором — к ее части EKS для вычисления $a_{\min}$.

В подпрограмме EKSTR находить сначала максимум и, если $x=1$, выходить из подпрограммы. Если же $x=0$, то после нахождения максимума находить минимум и их сумму. При обращении к части подпрограммы с именем EKS будет вычисляться только минимум.

3. ПРОГРАММИРОВАНИЕ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ ПЛ/1

Алгоритмический язык ПЛ/1 содержит основные элементы и свойства таких языков высокого уровня, как АЛГОЛ-60, ФОРТРАН, КОБОЛ. Он предназначен для программирования задач различных классов (выполнения научно-технических и инженерных расчетов, обработки экономической информации, решения задач логических, моделирования, в реальном масштабе времени и для разработки систем программного обеспечения ЭВМ).

Далее будет рассмотрено подмножество ПЛ/1 ОС ЕС, достаточное для практического программирования широкого класса задач обработки данных. Допустимые сокращения использованных ниже ключевых слов языка ПЛ/1 (указаны в скобках) следующие: BINARY (BIN); CHARACTER (CHAR); COLUMN (COL); DECIMAL (DEC); DECLARE (DCL); DEFINED (DEF); EXTERNAL (EXT); PROCEDURE (PROC).

3.1. ПРОСТЕЙШИЕ КОНСТРУКЦИИ ЯЗЫКА

К простейшим конструкциям языка относятся константы, переменные, функции и выражения, образуемые непосредственно из основных символов языка, составляющих его алфавит: все прописные латинские буквы, все арабские цифры; специальные символы: «__»; «=»; «+»; «—»; «*»; «.»; «%», «¬»; «|»; «>»; «?»; «/»; «(»; «)»; «,»; «;»; «:»; «'»; «&»; «_»; «<»; «$»; «@»;

и знак прямой решетки.

Константы

В языке ПЛ/1 можно использовать несколько типов констант.

Десятичные константы. Десятичные константы с фиксированной точкой (DECIMAL FIXED). Они записываются в виде целой и дробной части, разделенных десятичной точкой. Например, 329.443; —19.2; —6420.001; 85.49; .829; —3,5. Незначащие нули можно опускать.

При отсутствии десятичной точки константа называется целой десятичной константой, например —323; 84; —667;

Максимальная длина десятичной константы с фиксированной точкой, включая нули, составляет 15 десятичных цифр. Поэтому разрешается использовать числа x из интервала $-10^{15}<x<10^{15}$.

Десятичные константы с плавающей точкой (DECIMAL FLOAT). Эти константы используются для записи числовых значений, представленных в экспоненциальной форме. Так, число $81000000=81\cdot10^6$ компактнее можно записать как 81E6. Часть записи перед E называется *мантиссой*, а после E — *порядком*. Мантиссу можно задавать как десятичное число с фиксированной точкой, а порядок — как целое. Количество цифр мантиссы должно быть не более 16 цифр, а порядка — не более двух цифр, что определяет диапазон используемых чисел $10^{-78}<|x|<10^{75}$. Например,

значение числа	запись на языке ПЛ/1
0,00045	45E—5
$-29{,}8\cdot10^{-8}$	—29.8E—8
$12\cdot10^{8}$	12E8
—143000000000	—143E9
$0{,}5\cdot10^{-7}$	0.5 E—7 .5E—7

Точность константы с плавающей точкой задается числом p, которое равно количеству цифр мантиссы, включая нули. Например, для констант 8.3E8; 0.193E4; 0.89350E0 точность соответственно равна (2); (4); (6).

Точность констант с фиксированной точкой задается числами p и q, где p—общее количество цифр в записи константы, q — количество цифр в дробной части, т. е. справа от десятичной точки. Например, точность констант 08; 8; 8.0; .08; 0.08 соответственно равна (2,0); (1.0); (2.1); (2,2); (3,2).

▲ 3.1. Указать, какие из приведенных ниже записей можно рассматривать как а) десятичные константы с фиксированной точкой, б) плавающей точкой, в) неверные записи констант:

1) 85.; 2) 85; 3) .85; 4) —85.0; 5) 100^3; 6) +5; 7) $-5 \cdot 10^3$; 8) 200,2; 9) —3.01; 10) 966E0; 11) 4.3E12; 12) —1E.3; 13) E3; 14) 3.E0.5; 15) 1,2E8; 16) 1E1E1; 17) —8E.3; 18) 4E129; 19) 50.1E8; 20) —E+3; 21) 83.2E+6.

3.2. Записать приведенные ниже числа в виде десятичных констант с фиксированной точкой, состоящих из четырех цифр: 1) —3,01; 2) $(0,8)^2$; 3) $0,59 \cdot 10^2$; 4) $(-1)^3 \cdot 128$; 5) $+419 \cdot 10^1$; 6) $(-5)^3$; 7) $5,3 \cdot 10^{-1}$; 8) $0,68 \cdot 10^{-3}$; 9) $0,251 \cdot 10^3$.

▲ **3.3.** Представить приведенные ниже числа в виде десятичных констант с плавающей точкой в форме нормализованной мантиссы из трех цифр и порядка: 1) 25; 2) 10^6; 3) $343 \cdot 10^8$; 4) 0,000529; 5) —297,375256; 6) $8971,6751 \cdot 10^{-6}$; 7) 1980; 8) $0,257 \cdot 10^{-5}$; 9) $-197813 \cdot 10^8$; 10) $2/3 \cdot 10^{12}$; 11) 1/3; 12) 10^{-4}.

Двоичные константы. Двоичные константы с фиксированной точкой (BINARY FIXED). Константы данного типа представляют собой набор нулей и единиц. Дробная часть константы отделяется от целой точкой. Признаком двоичной константы является буква B в крайней правой позиции. Например, двоичным константам 101B; —101.01B; .0101B соответствуют десятичные эквиваленты $5; -5\frac{1}{4}; \frac{5}{16}$.

Максимальное число цифр двоичной константы с фиксированной точкой составляет 31.

Двоичные константы с плавающей точкой (BINARY FLOAT). Данные константы записываются в виде мантиссы и порядка, разделенных буквой E. Мантисса задается как двоичное число с фиксированной точкой. Порядок, указанный после буквы E, представляет собой десятичную степень двойки, на которую умножается мантисса для получения истинного значения. Запись кончается буквой B. Например, константа 110.1E4B соответствует числу 110.1B, умноженному на 2^4, т. е. $6\frac{1}{2} \cdot 2^4$, а константы 1110.01E—2B и —0.11E—3B соответственно числам $14\frac{1}{4} \cdot 2^{-2}$ и $-\frac{3}{4} \cdot 2^{-3}$.

Мантисса должна содержать не более 53 двоичных цифр, а порядок — не более 3 десятичных цифр, что определяет диапазон представления чисел от 2^{-209} до 2^{252}.

Строковые константы. В языке ПЛ/1 различают два типа строковых констант — символьные и битовые.

Символьные константы. Это произвольная последовательность буквенно-цифровых символов, заключенная в апострофы. Апостроф внутри константы задается двумя смежными апострофами. Например, 'СУММА ___ РУБ.'; 'ВИД'; 'ОБ"ЕКТ'; 'TIME'; 'ШИФР—2052'; 'AAAAA' эквивалентно (5)'A'; 'A1B1A1B1 A1B1' эквивалентно (3)'A1B1'.

Символьная константа может иметь любую длину, не превышающую 32767 символов.

Битовые константы. Это последовательность нулей и единиц, заключенная между апострофами. За закрывающим апострофом должна быть записана буква B. Например, '10100101'B; '001111000'B; '(3)01'B эквивалентно '010101'B. Максимальная длина битовой константы 32767 битов.

▲ **3.4.** Указать: а) какие из приведенных ниже записей являются двоичными константами с фиксированной точкой; б) представить эти константы в виде десятичных чисел: 1) 101B; 2) 101; 3) —1.5B; 4) 302B; 5) 111B; 6) —1010B; 7) '10'B; 8) —1.1B; 9) —0.1B; 10) —1.3B; 11) 0,0B; 12) 1.11E2.

▲ 3.5. Указать, какие из приведенных ниже записей можно рассматривать как символьные константы: 1) 'ABC'; 2) 'AB ___'; 3) 'AB'+3; 4) —'ABC'; 5) '(15)ABC'; 6) (15'ABC')

Переменные

Имена (идентификаторы) переменных, массивов, функций представляют собой последовательность буквенно-цифровых символов, начинающихся с буквы. Максимальная длина имени — 30 символов. Например, X; S; OLIMP; PSI; Y1; SUMMA; DATA; NUMER.

Различают переменные простые и с индексами (см. 2.1).

Простые переменные. Эти переменные обозначаются только именем. Например, X, ALPHA, Q1, A35.

Переменные с индексами. Данные переменные являются элементами массива и обозначаются индентификатором массива, за которым в круглых скобках через запятую указываются индексы. Например, A(5) — пятый элемент одномерного массива A; X(I, J) — элемент матрицы X, лежащий на пересечении i-й строки и j-го столбца.

Максимальное число индексов, допустимое в языке ПЛ/1, равно 32.

Индексы, указываемые в скобках, могут представляться целыми числами, переменными и арифметическими выражениями. При задании арифметического выражения в качестве значения индекса берется целая часть от значения вычисленного выражения. Например, если $a=8{,}2$ $b=0{,}4$, то при обращении к элементу массива ALPHA (A*B) будет выбрано значение третьего элемента массива ALPHA (3).

▲ 3.6. Указать, какие приведенные ниже записи являются идентификаторами переменных: 1) ABC; 2) 4AB; 3) SUM; 4) A/BC; 5) LUM111B; 6) 'SQ706'; 7) IBM/360; 8) A*B; 9) α(1,2); 10) X(1597); 11) N(I, J*K); 12) B(—5, 8, 0).

3.7. Составить из символов X и Y всевозможные идентификаторы длиной в три символа.

3.8. Указать причины неверной записи следующих идентификаторов: 1) AE7·4; 2) DM—101; 3) 8ALPHA; 4) ПОТОК 1÷3; 5) B__ 10; 6) &A3.

Описание типа переменных. Переменные, так же как и константы, могут быть десятичными или двоичными, символьными или битовыми.

Для описания типа переменных служит оператор описания данных DECLARE.

Формат этого оператора

DECLARE <имя переменной> <совокупность описателей>.

Набор описателей определяется типом переменных. Например, переменную x, принимающую значения в диапазоне от 0 до 200 000, можно объявить

DECLARE X DECIMAL FIXED (6);

Описание переменных арифметического типа. Характеристика описателей этого типа данных представлена в табл. 3.1, совокупность описателей в зависимости от типа переменных — в табл. 3.2.

Опущенные в операторе DECLARE описатели принимаются по *соглашению (умолчанию)* автоматически из совокупности описателей. Если для переменной не указаны никакие описатели в операторе DECLARE, то по соглашению приписываются описатели по первой букве имени: именам, начинающимся с букв I, J, K, L, M, N, — описатель FIXED BINARY(15); именам, начинающимся с любой другой буквы, — описатель DECIMAL FLOAT (6).

Для значений разрядности (p, q) по соглашению принимаются значения, приведенные в табл. 3.3.

Таблица 3.1

Характеристики	Запись описателя в операторе	Описатель по соглашению
Система счисления:		
десятичная	DECIMAL	DECIMAL
двоичная	BINARY	
Способ представления:		
с фиксированной точкой	FIXED	
с плавающей точкой	FLOAT	FLOAT
Тип переменной:		
вещественный	REAL	REAL
комплексный	COMPLEX	
Разрядность (точность представления значения переменной):	(*p*) или (*p*, *q*)	См. табл. 3.3
количество значащих цифр	*p*	
длина дробной части	*q*	

Таблица 3.2

Тип переменных	Совокупность описателей	Примечание
Десятичные с фиксированной точкой	DECIMAL FIXED (*p*, *q*)	p — общее количество цифр, включая цифры дробной части $1 \leqslant p \leqslant 15$; q — количество цифр дробной части $0 \leqslant q \leqslant p$
Целые десятичные	DECIMAL FIXED (*p*, 0) или DECIMAL FIXED (*p*)	$1 \leqslant p \leqslant 15$
Двоичные с фиксированной точкой	BINARY FIXED (*p*, *q*)	$1 \leqslant p \leqslant 31$ $0 \leqslant q \leqslant p$
Двоичные целые	BINARY FIXED (*p*)	$1 \leqslant p \leqslant 31$
Десятичные с плавающей точкой	DECIMAL FLOAT (*p*)	p — количество цифр мантиссы $1 \leqslant p \leqslant 16$
Двоичные с плавающей точкой	BINARY FLOAT (*p*)	$1 \leqslant p \leqslant 53$

3.9. Определить, какие характеристики будут присвоены переменным с помощью оператора

```
DECLARE  (X, Y) DECIMAL FIXED (8,2),
         Z FLOAT (3) DECIMAL,
         (A, B, C) FIXED (7,1),
         T1 BINARY (25),
         (Q1, Q2) FIXED BINARY (18,6);
```

Таблица 3.3

Описатели переменных	Значение разрядности	
	по соглашению	максимальное
DECIMAL FIXED	(5,0)	(15, q)
DECIMAL FLOAT	(6)	(16)
BINARY FIXED	(15,0)	(31, q)
BINARY FLOAT	(21)	(53)
CHARACTER		32 767
BIT		32 767

▲ **3.10.** Указать, какие описатели будут присвоены переменным с помощью оператора

```
DECLARE  R FIXED,
         L BINARY (6),
         (E, F) FIXED (3),
         (G, H) DECIMAL (4);
```

▲ **3.11.** Определить, какие описатели будут присвоены переменным в указанном фрагменте программы:

```
DECLARE (BETA, GAMMA, MIN, MAX) DFC;
        . . . . . .
        SUMMA=0; N=2*4.65;
        I=10.3;
```

Описание строковых переменных. Для описания строковых переменных применяются описатели типа CHARACTER (символьные данные) и типа BIT (битовые данные) совместно с описателем длины, задаваемым в форме (l), где $1 \leqslant l \leqslant 32767$.

● **3.12.** Указать, какие описатели присваиваются переменным с помощью оператора

```
DECLARE  AH CHARACTER (150),
         BH CHARACTER (64),
         S1 BIT (48);
```

Значениями переменных AH, BH могут быть символьные константы с числом символов (длиной) соответственно 150 и 64, а переменной S1 — битовые константы длиной 48 битов.

Массивы

Массив — множество последовательных величин, названных одним именем. Каждая отдельная величина является *элементом массива* и записывается как переменная с индексом.

Для описания массивов и резервирования памяти под массивы используется оператор DECLARE.

Этот оператор имеет следующий формат:

DECLARE < имя массива > < описатель разномерности > < совокупность описателей >;

В описателе размерности указываются границы изменения каждого из индексов массива. Если индекс изменяется от 1 до некоторой величины, то в описателе размерности допускается указывать только его верхнюю границу. Например, описатель размерности одномерного массива в 50 элементов с изменением индекса от 1 до 50 можно записать (1 : 50) или (50). Если индекс изменяется от 0 до 49, то описатель размерности этого же массива следует записать (0 : 49), т. е. указать нижнюю и верхнюю границы изменения индекса.

Допускается работа с массивами, индексы которых имеют отрицательное значение.

3.13. Определить размерности массивов, объявленных с помощью оператора

```
DECLARE  A(100) FIXED DECIMAL (8,2),
         B(5, 10) FLOAT (3),
         C(0 : 2, 0 : 6, 0 : 7) BINARY,
         (X, Y) (10)FIXED (2),
         D(—3 : 26) BIT (1);
```

3.14. Описать приведенные ниже переменные и массивы с помощью операторов описания данных:

1) r и q — вещественные десятичные с фиксированной точкой и точностью 11 цифр, из них 3 дробные цифры; 2) x, y — вещественные десятичные с плавающей точкой и точностью 8 цифр; 3) i, j, k — двоичные с фиксированной точкой и точностью 15 цифр; 4) A — целочисленный массив из 100 элементов (диапазон чисел от 1000 до 999999); 5) B — двумерный массив (10×10). Все элементы — вещественные десятичные целые числа в диапазоне от 100 до 2000.

▲ **3.15.** Описать переменную x, имеющую значения:

1) 2845; 2) —81.3E5; 3) 101.1B; 4) 101.1; 5) —1.11E5B; 6) 'ALPHA'; 7) (4) 'C'; 8) —30000; 9) 102.34E—6.

▲ **3.16.** Описать массивы: 1) X — одномерный с границами изменения индекса от 0 до 199, содержащий константы десятичные с фиксированной точкой и разрядностью (8, 2); 2) X — двумерный с границами изменения индексов от —2 до 7 по обоим измерениям, содержащий константы десятичные с плавающей точкой точностью (6); 3) X — трехмерный с границами изменения индексов от 0 до 9, —5 до 4, 1 до 3, содержащий константы символьного типа длиной 16 символов.

▲ **3.17.** Описать массивы, полагая, что все их элементы — целые десятичные числа с количеством цифр не более восьми: 1) (a_i), $i=1, 2, ..., 10$; 2) (x_k), $k=0, 1, 2, ..., 19$; 3) (d_i), $i=-8, -7, ..., 25$; 4) (a_{ij}), $i=1, 2, 3$, $j=1, 2, 3, 4$; 5) (x_{ijk}), $i=1, 2, 3$, $j=1, 2, 3, 4, 5$, $k=1, 2$; 6) (y_{ijk}), $i=0, 1, 2$, $j=0, 1, 2, 3$, $k=1, 2$.

Функции

В процессе решения задач часто возникает необходимость в вычислении значений элементарных функций (логарифма, корня квадратного, экспоненты, синуса и т. д.). Для этого надо написать наименование функции и в круглых скобках указать аргумент. Например, SIN(X), COS(X), LOG(X), LOG10(X) и т. д. Наиболее часто употребляемые функции приведены в табл. 3.4. Аргументами функций могут быть константы, арифметические выражения, переменные простые, с индексами, а также сами функции. Например,

запись математическая	запись на языке ПЛ/1
$\sqrt{e^x}$	SQRT (EXP(X))
$\sin(x+y)$	SIN (X+Y)
$\lvert\beta\rvert$	ABS (BETA)
$\ln\sqrt{\cos(a/d)}$	LOG (SQRT (COS (A/D)))
$e^{\sqrt{a_{ij}}}$	EXP (SQRT (A(I, J)))

Таблица 3.4

Функция	Запись математическая	Запись на языке ПЛ/1	Примечание
Синус	$\sin(x)$	SIN(X)	Аргумент в радианах
		SIND(X)	Аргумент в градусах
Косинус	$\cos x$	COS(X)	Аргумент в радианах
		COSD(X)	Аргумент в градусах
Арктангенс	$\operatorname{arctg} x$	ATAN(X)	Результат в радианах
		ATAND(X)	Результат в градусах
Тангенс	$\operatorname{tg} x$	TAN(X)	Аргумент в радианах
		TAND(X)	Аргумент в градусах
Экспонента	e^x	EXP(X)	
Логарифм натуральный	$\ln x$	LOG(X)	Аргумент больше 0
Логарифм десятичный	$\lg x$	LOG10(X)	Аргумент больше 0
Корень квадратный	$\sqrt{x}$	SQRT(X)	Аргумент больше или равен 0
Абсолютное значение	$\lvert x \rvert$	ABS(X)	—
Деление по модулю	$x - \left]\frac{x}{y}\right[\cdot y$	MOD (X, Y)	—
Максимальное значение	$\max\{x, y, d, z\}$	MAX (X, Y, D, Z)	Аргумент не должен быть массивом
Минимальное значение	$\min\{x, y, z, q\}$	MIN (X, Y, Z, Q)	Аргумент не должен быть массивом

Результат выполнения математических функций всегда имеет форму представления с плавающей точкой.

Выражения

Выражения в зависимости от используемых объектов разделяются на арифметические, логические и выражения над строками.

Арифметические выражения. Эти выражения записываются с использованием числовых констант, переменных, функций, знаков арифметических операций и круглых скобок. Значением арифметического выражения всегда является число.

В языке ПЛ/1 используются арифметические операции сложения, вычитания, умножения, деления, возведения в степень. Порядок выполнения операций в арифметическом выражении задается круглыми скобками. При их отсутствии установлен следующий порядок старшинства арифметических операций: сначала выполняются операции вычисления функций, затем операции возведения в степень, далее все операции умножения и деления и, наконец, все операции сложения и вычитания.

Операции одинакового старшинства выполняются в том порядке, в котором они следуют в выражении, т. е. слева направо. Исключение составляют операции возведения в степень, которые выполняются в порядке следования справа налево. Например,

запись математическая	запись на языке ПЛ/1
ax^2+b	A*X**2+B
x^{-3}	X**(—3)
$\sqrt{z}$	Z**(1./N)
$\frac{-8{,}3\cdot 10^3+a\cdot n+10^{-8}c}{e^x+d}$	(—8.3E3+A*N+1.E—8*C)/(EXP(X)+D)
$\frac{\sin x^2+\cos^2 x}{\lvert\tau\rvert}$	(SIN(X**2)+COS(X)**2)/ABS(TAU)

При делении целого на целое рекомендуется использовать функцию DIVIDE вместо операции деления. Функция DIVIDE(X, Y, p, [q]) позволяет получать результат деления X/Y с требуемой точностью p, q.

▲ **3.18.** Записать следующие арифметические выражения:

1) $\ln x + y^k x + 5a^2$; 2) $\dfrac{a\sqrt{\sin^2 x+\cos^2 x}-\lg^2(bx)}{\sqrt{a^3+b^3+c^3}}\cdot 10^{-5}$;

3) $\dfrac{e^{\sin x}+\sqrt[3]{\sin x}}{1-\ln(\sqrt{x^2-a}-0{,}82)}+0{,}25\dfrac{(x^2-a)\sin x}{\cos(x^2-a)+\sin x}$;

4) $\sqrt[3]{p^2}+\ln|s^3-1|-\sqrt{|s^3-1|}$.

▲ **3.19.** Записать следующие арифметические выражения:

1) $\dfrac{x}{1+x^2/(2y)}$; 2) $\dfrac{a+bx}{c+d}$; 3) $e^x-\sin^2 x$; 4) $1+\operatorname{arctg}\dfrac{x}{1+\sqrt{x}}$;

5) $ax^3+bx^2+a\cos^3|x|$; 6) $\dfrac{\sin^3 x-\cos x^3}{2x-10^{-3}}$; 7) $\dfrac{x^m-a^m}{\sqrt{ax}}$;

8) $e^{-ax}\sin(\omega t+\varphi)$.

▲ **3.20.** Записать следующие арифметические выражения:

1) $5\sin x+x^7\left(1+\dfrac{x-z/x}{x+z/x}\right)\sin x-1$; 2) $3{,}14r^2+v^2h+\dfrac{vh+1}{v}+3{,}14r^3$;

3) $(a+b)\sin a+(a-b)\cos b+\dfrac{\left[a-b\right]_a^2}{\sin a+\cos b}$; 4) $1+x+\dfrac{x^2}{2!}+\dfrac{x^3}{3!}$;

5) $1+\dfrac{x}{1+x/(1+x)}$; 6) $((x^a)^b)^c$; 7) $\dfrac{(x^y/y^x)^z}{zy}$.

● **3.21.** Указать, какие ошибки допущены в записи следующих арифметических выражений: 1) (Z*Y)*X); 2) —S*—17; 3) X**—3; 4) $\dfrac{A+B}{C}$.

1) Несоответствие скобок; 2) , 3) недопустимы рядом два знака операций; 4) недопустимы подстрочные символы (вся запись выражения в строку).

● **3.22.** Указать, эквивалентны ли следующие пары записей арифметических выражений:

X*A/C*D	((X*A)/C)*D
—F**2	—(F**2)
X**2**A	X**(2**A)

Да, эквивалентны с учетом старшинства операций.

▲ 3.23. Определить порядок выполнения операций:

1) $A^{1)} * 3.27^{2)} - Z^{3)} + D$; 2) $Y^{1)} ** (N^{2)} - A)^{3)} ** X^{4)} + I^{5)} / (-(Y^{6)} + I)^{7)} ** I^{8)}$;

3) $-Z^{1)} * X^{2)} / I^{3)} * J^{4)} / K^{5)} * R^{6)}$; 4) $M^{1)} + N^{2)} / (M^{3)} + N)^{4)} * K^{5)} / (2.3^{6)} + X)^{7)} - (M^{8)} - N)^{9)} / Z$.

Знаки арифметических операций обозначены цифрами 1, 2, 3, ...

Логические выражения. Простейшее логическое выражение — это отношение типа $a<b$ (меньше); $a>b$ (больше); $a=b$ (равно); $a \neg < b$ (не равно); $a \neg > b$ (не больше); $a \neg = b$ (не меньше); $a<=b$ (меньше или равно); $a>=b$ (больше или равно), где a и b — операнды. Например,

запись математическая	запись на языке ПЛ/1
$a>100$	A>100
$b^2-4ac>0$	B**2—4*A*C>0
$i \neq j$	I¬ =J
$\sqrt{x} \geqslant 2{,}83$	SQRT(X)>=2.83

Результат вычисления логического выражения — строка битов единичной длины. Если проверяемое отношение истинно, то результат равен '1'B; если ложно, то '0'B.

В зависимости от операндов различают сравнения: алгебраические (означающие сравнение числовых величин с учетом знаков); символьные (означающие последовательное попарное сравнение символов слева направо; более короткая строка символов дополняется справа пробелами) и битовое (означающее сравнение двоичных цифр слева направо; более короткая строка бит дополняется справа нулями).

▲ **3.24.** Определить результат выполнения операций сравнения 1) A>B; 2) A<=B; 3) H<=F; 4) U1<U2; 5) U1 ¬=U2 над следующими данными: $a=83.25$; $b=29.38$; $h='ABCDE'$; $f='ABC'$; $v_1='01101011'$, $v_2='101'$, описанными оператором

```
DECLARE  A DECIMAL FIXED (8,3),
         B DECIMAL FIXED (8,3),
         H CHARACTER (8),
         F CHARACTER (5),
         U1 BIT (8);
         U2 BIT (3);
```

▲ **3.25.** Записать логические выражения, при которых можно вычислить следующие арифметические выражения: 1) $z=1/(x^5y^3)$ 2) $x=(b+\sqrt{b^2-4ac})/(2a)$ при $a \neq 0$; 3) $z_i=b_i \ln x_{ij}$; 4) $y=a \lg x$.

Логические операции. К этим операциям относятся операции отрицания (¬), умножения (конъюнкция &), сложения (дизъюнкция |), выполняемые над данными типа строки битов последовательно над каждой парой битов. Результат логических операций — строка битов длиной, равной длине большего операнда. Правила выполнения логических операций задаются таблицей истинности (табл. 3.5).

Кроме рассмотренных операций существует еще операция сцепления ||, назначение которой — соединение операндов в одну строку.

Например, 'ABC'||'DE' дает 'ABCDE' (сцепление строк символов), '1001' B||'101'B дает '1001101'B (сцепление строк битов).

Таблица 3.5

A	B	¬ A	A | B	A&B
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	0

Соотношение старшинства арифметических и логических операций следующее: 1) возведение в степень (**), отрицание ( ); присвоение знака (+), (—); 2) умножение (*), деление (/); 3) сложение (+), вычитание (—); 4) сцепление (||); 5) сравнение; 6) логическое умножение (&); 7) логическое сложение (|).

▲ **3.26.** Определить результат логических операций 1) ¬ A; 2) ¬B; 3) A|B; 4) A| ¬ B; 5) B&C; 6) B||C; 7) F||H над следующими данными: a='1011'B; b='01110'B; c='11101000'B; f='ALFA'; h='PLUS', описанными оператором

```
DECLARE  A BIT (4),
         B BIT (5),
         C BIT (8), (H, F) CHARACTER (6);
```

▲ *3.27.* Определить значение следующих логических выражений: 1) X>0&Y>0 при $x=5$, $y=-1$; 2) X>0|Y>0 при $x=0$, $y=2$; 3) X=0| ¬ Y ¬=0 при $x=1$, $y=-2$; 4) ¬ A|X&D<C при a='0'B, x='1'B, $d=5.5$, $c=3.8$; 5) (¬X&Y|Z)&'1'B при x='0'B, y='1'B, z='0'B.

▲ **3.28.** Записать:

а) логические выражения, при которых можно вычислить следующие арифметические выражения: 1) $z=ae^x/(c\ln y)$; 2) $z=\ln y/(x-y)$; 3) $z=\sqrt{x/(\ln y+x)}$; 4) $z=1/[a+1/(b+c)]$; 5) $z=a/(bc)$;

б) условие кратности трем для целочисленной переменной n;

в) условие кратности пяти одновременно двух целочисленных переменных k, l.

▲ **3.29.** Определить порядок выполнения операций

1) $A\overset{1)}{*}\neg B\overset{2)}{-}C\overset{3)}{|}D\overset{4)}{}$; 2) $B\overset{1)}{\&}A\overset{2)}{\&}C\overset{3)}{|}B\overset{4)}{|}\neg C\overset{5)}{}$; 3) $A\overset{1)}{+}Z\overset{2)}{\|}C\overset{3)}{\&}AK\overset{4)}{+}+\neg X\overset{5)}{|}Y\overset{6)}{}$; 4) $\neg X\overset{1)}{|}Y\overset{2)}{\&}Z\overset{3)}{}$.

Знаки арифметических и логических операций обозначены цифрами 1, 2, 3 и т. д.

3.2. ВВОД И ВЫВОД ДАННЫХ

Ввод и вывод значений простых переменных

Ввод — вывод данных потоком (STREAM) возможен в трех режимах, в которых ввод — вывод управляется соответственно списком (режим LIST), редактированием (режим EDIT), данными (режим DATA).

Ввод — вывод данных реализуется с помощью операторов ввода GET и вывода PUT. Ниже рассматривается организация ввода данных с перфокарт (со стандартного входного файла SYSIN) и вывода данных на широкую печать (на стандартный выводной файл SYSPRINT).

Ввод и вывод в режиме LIST. Форматы операторов ввода и вывода

GET LIST (список переменных);

PUT LIST (список переменных);

Список переменных указывает, каким переменным нужно присвоить вводимые значения или значения каких переменных требуется вывести.

При вводе данные с вводного устройства считываются символ за символом и рассматриваются как строки символов. Границы перфокарт игнорируются. Концом значения переменной во входном потоке является пробел или запятая. Если во входном потоке (на перфокартах) встречаются две запятые подряд (или между ними есть пробелы), то значение текущего элемента считается пустым и присваивания значения не происходит. На перфокартах числа представляются в виде десятичных констант, строки символов и битов — в виде последовательности символов, заключенной в апострофах и символа B (для строк битов). Например, если входной поток имеет вид ␣␣23,7␣28␣␣ „ —130, то при выполнении операторов

DECLARE (X, Y, Z, A, C) FIXED (5,0);

GET LIST (X, Y, Z, A, C);

переменным будут присвоены значения $x=00023$; $y=00007$; $z=00028$; для a присваивание пропускается; $c=-00130$.

При выводе данные выводятся строго в соответствии со списком переменных в операторе PUT LIST в форме констант, вид которых определяется характеристиками соответствующего значения. Символьные константы в апострофы не заключаются. Битовые константы выводятся в апострофах. Например,

выводимое значение	описатель переменной	вид строки печати
+85	FIXED(2)	$w=p+3$␣␣85
—85	FIXED(2)	␣␣85
—.05	FIXED(2,2)	$w=p+3$—0.05
—21.7	FIXED(3,1)	␣—21.7
—25E6	FLOAT(2)	$w=p+6$—2,5E—06
.151E—3	FLOAT(3)	$w=p+6$␣1.51E—03
'ABC'	CHARACTER(3)	ABC
'101'B	BIT(3)	'101'B

В списке выводимых данных могут стоять константы, переменные, выражения. При выводе данные разделяются пробелами и размещаются на листинге печати с позиций 1, 25, 49, 73, 97. Например, при выполнении оператора

PUT LIST (15E—3, —3.2, 2, 1.11B);

будет напечатано

␣1.5 E—02␣'3.2 ␣␣␣2␣␣ 1.7

▲ **3.30.** Указать, какие значения получат переменные x, a, b при выполнении оператора

```
DECLARE (X, A, B, C, D) FIXED (5,1);
GET LIST (X, A, B, C, D);
```

если входной поток имеет вид

```
__25.3,__0.1,___,___,___—18.239__2384.2___
```

▲ **3.31.** Указать, какие значения получат переменные при выполнении следующих операторов программы:

```
DECLARE          A FLOAT (5);
                 B FIXED (8, 0),
                 C CHARACTER (5), D CHARACTER (6), F BIT (5);
GET LIST (A, B);
. . . . . .
GET LIST (C, D);
GET LIST (F);
```

если входной поток имеет вид

```
7.02,__25,'ABCD'___'ПРОБЛЕМА'__'1011'B,
```

▲ **3.32.** Описать переменные x, y, z и указать вид входного потока для ввода в режиме LIST числовых значений переменных —209; 353.429; 0.000000028.

▲ **3.33.** Указать, какие значения будут присвоены переменным после выполнения операторов

```
1)  DECLARE          (A, B, C) FIXED (5,3),
                     (E, F, D) FIXED (7,4),
                     (P, Q) CHARACTER (13);
    GET LIST (A, B, C);
    GET LIST (E, F, D);
    GET LIST (P, Q);
2)  GET LIST (A, B);
    GET LIST (E, F);
    GET LIST (C, D);
```

если входной поток имеет вид

```
— 138__2935__179693___,___,___,___,___,...
___,___— 010005___,___,___2198.0___,___...
___'ТЕМПЫ__РОСТА'__'ГОДОВОЙ__ДОХОД'__
```

В варианте 2 отсутствует оператор DECLARE.

● **3.34.** Указать вид строки печати при выводе следующих значений переменных:

описатель переменной	выводимое значение	вид строки печати
FIXED (4)	—250	___,___,___— 250
FIXED (3)	0	___,___,___0
FIXED (5,3)	0	___,___,___0.000
FLOAT (5)	—0.00065	—6.5000E—04
BINARY FIXED (3)	11B	___,___,___3
CHARACTER (16)	'ОЛИМПИАДА—80'	ОЛИМПИАДА—80
BIT (3)	'101'B	'101'B

● 3.35. Указать вид печати при выполнении следующих операторов:

```
DECLARE, X FIXED (5,2), Y FLOAT (2),
         A CHARACTER (6), S BINARY (4),
         Z BIT (3);
X=—2,834; Y=0.0139; A='ИТОГО'; Z='111'B;
S=101B;
PUT LIST (X, Y, A, S);
. . . . . .
PUT LIST (Z, X, '␣', '␣', A);
```

После выполнения первого и второго оператора PUT будет напечатано:

␣.␣.␣—2.83	␣1.4E—0,2	ИТОГО	␣.␣.␣5	'111'B
␣.␣.␣—2.83			ИТОГО	

▲ 3.36. Указать вид печати при выполнении следующих операторов:

```
DECLARE SUM FIXED (8,2), Z FLOAT (4);
SUM = — 318.253; Z = 10 003.29;
PUT LIST ('СУММА=', SUM,'ПРОИЗВЕДЕНИЕ=', Z);
```

▲ 3.37. Указать вид печати при выполнении операторов

```
PUT LIST (A, '␣', B);
PUT LIST (E, D, '␣',F) SKIP;
```

для данных задачи 3.33 (вариант 2). Описатель SKIP указывает на печать с начала новой записи (строки печати).

▲ 3.38. Указать вид печати при выполнении следующих операторов:

```
1) PUT LIST (29.83, 'БАЛАНС=', 43E—8);
2) DECLARE A FIXED (2,0), B FIXED (4,0);
       A=—25; B=410;
   PUT LIST (A+B, '␣', A—B);
3) DECLARE (I, J) CHARACTER (8), (T, F) FIXED (7,3);
   T=999.11; F=0.2897E1;
   I='ПЕРИОД='; J='ФУНКЦИЯ=';
   PUT LIST (I, T, J, F)
```

3.39. Дать описание и записать операторы вывода для переменных, имеющих значения 1) SU=1; POL=120; R=—829.48; S1=—0.00038; 2) X=—25.004; Y=170.749; Z=1.7E19; 3) A='15␣ЧАС␣0,2 МИН'; B='СРЕДА'; 4) E=0,873E—13; F=1101B; U='1010'B.

Ввод и вывод данных в режиме EDIT. Ввод информации с перфокарт в данном режиме осуществляется с помощью оператора

```
GET EDIT (список переменных) (список форматов);
```

а вывод на широкую печать — оператором

```
PUT EDIT (список переменных) (список форматов);
```

Список переменных задается аналогично заданию списка переменных в режиме LIST.

Список форматов представляет собой последовательность форматов, отделенных запятыми: формат 1, формат 2, ..., формат *n*. Форматы данных указывают, из

скольких символов и в какой форме должно быть представлено значение очередной переменной при вводе или выводе. Ввод — вывод значений переменных осуществляется строго в соответствии со списком переменных и списком формата: первая переменная связывается с первым форматом данных, вторая — со вторым и т. д. Если список переменных больше списка форматов, то список форматов просматривается с начала. Если список форматов больше списка переменных, то конец списка форматов игнорируется.

Наиболее употребимы следующие форматы данных:

Формат типа F. Это формат для ввода — вывода данных с фиксированной точкой.

Общий вид формата

aF(w) или aF(w, d),

где w — длина поля, занимаемая константой; d — количество цифр дробной части; a — коэффициент повторения формата.

Например, F(7) при вводе — выводе константа займет поле в семь позиций; F(8,3) — константа займет поле в восемь позиций, из них три позиции отводится под дробную часть; 3F(5) означает, что надо ввести (вывести) три константы по формату F(5). Например,

формат данных	выводимое значение	вид строки печати
F(6)	—189	␣␣␣ — 189
F(6)	0	␣␣␣␣␣0
F(6,2)	0	␣␣0.00
F(7,3)	—0,05	␣ — 0.050
F(7,2)	129.392	␣129.39
F(8,5)	4752.1	* * * * * * * * (число в данном формате не умещается)

Формат типа E. Это формат для ввода — вывода данных с плавающей точкой.

Общий вид формата

$aE(w, d)$ или $aE(w, d, s)$,

где w — ширина поля, занимаемого числом; d — количество цифр дробной части мантиссы; s — количество значащих цифр.

Если параметр s не установлен, то считается $s=d+1$. Если параметр d при выводе равен 0, то точка и дробные цифры не выводятся.

выводимое значение	формат данных	вид строки печати
—3.14159	E(12,4)	␣␣␣ — 3.1416E␣00
—3.14159	E(12, 4, 5)	␣␣␣ — 3.1416E␣00
—3.14159	E(12, 4, 6)	␣ — 31.4159E — 01
—3.14159	E(12, 4, 7)	— 314.1590E — 02
—3.14159	E(12, 0, 5)	␣␣␣—3.1416E—04

Формат типа A. Этот формат используется для ввода — вывода строковых данных.

Общий вид A(w).

Это означает, что вводится (выводится) последовательность из w символов. Если длина вводимого (выводимого) поля $l<w$, то происходит расширение данного пробелами справа, если же $l>w$, то выполняется усечение лишних символов. Если w не задано, то вводится (выводится) фактическое изображение символьных данных. Например, если переменным X, B символьного типа присвоены значения X='ABCDE'; B='TEKCT'; то строка печати после выполнения оператора

PUT EDIT (X, B) (A (3), A);

имеет вид ABCTEKCT.

При выполнении оператора

```
PUT EDIT (X, B) (A (8), A);
```

строка печати имела бы вид ABCDE␣␣␣␣␣ТЕКСТ

Управляющие форматы. Эти форматы используются для пропуска части символов при вводе и расстановке пробелов при выводе.

Формат X *(w)*. Этот формат при вводе означает пропуск *w* символов, при выводе — пропуск *w* позиций.

Формат SKIP *(w)*. Данный формат указывает пропуск *w* записей, считая текущую запись (запись при вводе — одна перфокарта, при выводе — одна строка печати). Например, SKIP(1) или SKIP вызывает переход к следующей записи, SKIP(2) вызывает двойной интервал между строками при выводе.

Формат COLUMN *(w)*. Этот формат используется для установки номера позиции при вводе — выводе. Например, по оператору

```
PUT EDIT(X) (COLUMN (10), F(5));
```

значение *x* будет печататься с десятой позиции строки.

Управляющие форматы в конце списка форматов не воспринимаются.

▲ **3.40.** Указать вид печати при выполнении оператора

```
PUT EDIT (X, Y, X1, Y1) (SKIP, F (8,3), X(10), F(8,3));
```

если выводимые переменные имеют следующие значения: X1 = =84.038; Y1=164.501; X=123.051; Y=—84.269.

Вид печати:

␣123.051 ␣␣␣␣␣␣␣␣␣␣ ␣—84.269

F (8,3) 10 пробелов F (8,3)

␣␣84.038 ␣␣␣␣␣␣␣␣␣␣ ␣164.501

F (8,3) 10 пробелов F (8,3)

Значения *x*, *y* выводятся на первую строку печати. Список переменных не исчерпан, поэтому просмотр списка форматов повторяется с самого начала.

▲ **3.41.** Определить, какие значения будут присвоены переменным, и вид печати при выполнении операторов

```
GET EDIT (X, Y, Z) (COLUMN (4), 3F (3));
GET EDIT (A, B) (2F (4,1));
PUT EDIT (A, B) (COLUMN (5), F (4,1), COLUMN (25), F (4,1));
```

если входной поток имеет вид

123456␣␣83␣␣2982. 1 — 1. 3␣␣␣␣␣

▲ **3.42.** Составить оператор вывода, управляемый редактированием, для печати заголовка в виде

ТАБЛИЦА␣␣ВЫЧИСЛЕНИЯ␣␣ПОЛИНОМА

X Y

При выводе на печать отступить от края листа на 10 позиций. Оператор вывода будет иметь вид

```
PUT EDIT ('ТАБЛИЦА␣␣ВЫЧИСЛЕНИЯ␣␣ ПОЛИНОМА', (27)'—','X', 'Y')
```

SKIP, X (10), A, SKIP (0), X (10), A SKIP, X (17), A, X (10), A);

печать 1 строки — подчеркивание в той же строке — печать второй строки

▲ **3.43.** Определить, какие значения будут присвоены переменным после выполнения оператора

GET EDIT (A, B, C) (F (3), COLUMN (2), F (3), SKIP (2), F (2));

если входной поток имеет вид

␣␣1␣␣␣␣␣␣␣␣␣...

2 3 4 5␣␣␣␣␣␣␣␣...

6 7 8 9 A B␣␣␣␣␣␣...

1 2 3 4 5 6 7␣␣␣␣␣...

▲ **3.44.** Составить оператор вывода, управляемый редактированием, для печати значения переменной z:

	выводимое значение	вид строки печати
1)	829	␣␣␣829
2)	—13.829	␣␣␣—13.83
3)	—13.829	—1.38E␣␣01
4)	—2551	Z=—2551

▲ **3.45.** Указать, какие значения будут присвоены вещественным переменным a и b после выполнения оператора:

1) GET EDIT (A,B) (F (4));

если входной поток имеет вид 12345678␣␣␣␣. . .

2) GET EDIT (A, B) (F (3,2), F (4,2));

если входной поток тот же;

3) GET EDIT (A, B) (F (5,2), F (5,3));

если входной поток имеет вид 501.239.1␣␣␣␣. . .

4) GET EDIT (A, B) (E (5,1));

если входной поток имеет вид 1234567E—2␣␣␣␣. . .

5) GET EDIT (A, B) (E (6,2));

если входной поток имеет вид 2.E9␣␣␣.749␣␣␣␣. . .

▲ **3.46.** Указать вид строки печати при выполнении оператора:

1) PUT EDIT (A, A) (SKIP, A, COLUMN (10), A);

если значение символьной переменной a='РАСХОД';

2) PUT EDIT (X, Y) (SKIP, 2 (COLUMN (3), F(4), SKIP (2)));

если значения переменных $x=2990$ и $y=-5$;

3) PUT EDIT ('АРГУМЕНТ', 'ФУНКЦИЯ', X, Y)
(SKIP, A, COLUMN (20), A, COLUMN (1), F (7,3), COLUMN (20), F (7,3));

если $x=0.001$, $y=-8.29$.

▲ **3.47.** Написать операторы вывода на печать заголовков:

1)
ТАБУЛИРОВАНИЕ␣␣ФУНКЦИИ

X Y

2)
МАТЕМАТ. ОЖИДАНИЕ␣␣...␣␣ДИСПЕРСИЯ
(20 пробелов)

MX = ␣␣0.9995 DX = ␣␣8.9734E − 01

3)
КОРНИ␣␣УРАВНЕНИЯ

␣␣X1 = − 0.7777
␣␣X2 = − 0.2222

Ввод и вывод данных в режиме DATA. Ввод — вывод данных в этом режиме выполняется операторами:
ввода
GET DATA (список переменных);
вывода
PUT DATA (список переменных);

При вводе поток должен содержать вводимые данные в форме операторов присваивания, отделенных друг от друга запятыми или пробелами. Например,
X=5␣␣Y=3.297␣␣Z=−32.18, D=1E−06;

Конец входного потока, вводимого одним GET, отмечается точкой с запятой. Порядок следования переменных в списке может быть произвольным.

Вывод данных выполняется в форме операторов присваивания в последовательности, соответствующей списку переменных. Вид констант такой же, как и при выводе, управляемом списком. Символьные константы заключаются в кавычки. Данные располагаются в строке печати начиная с тех же позиций, что и при выводе, управляемом списком, т. е. с позиций 1, 25, 49, 73, 97.

▲ **3.48.** Указать, какие значения будут присвоены переменным после выполнения операторов

```
DECLARE (A, B, C) FIXED (6,2),
        F FLOAT (4), I CHARACTER (5);
GET DATA (A, B, C, F, I);
```

а входной поток имеет вид
A = 899.35, F = 1. 5E − 6, B = − 29.03␣␣␣␣I = 'NAME';

● **3.49.** Указать вид печати после выполнения операторов

```
DECLARE (A, B, C) FIXED (5, 2),
        Z FIXED (3), H CHARACTER (6);
A = − 13.45; B = − 0. 823;
C = 199.38; Z = − 113; H = 'СУММА';
PUT DATA (A, B, C, Z, H) SKIP;
```

Вид печати:
A = ␣␣␣␣− 13.45 B = ␣␣␣␣␣␣− 0.82 C = ␣␣␣␣ 199.38;
Z = ␣␣␣␣ − 113 H = 'СУММА␣␣'

▲ **3.50.** Указать, какие значения будут присвоены переменным после выполнения следующих операторов:

```
DECLARE (X, Y, Z) FIXED (5,2), W CHARACTER (8);
GET DATA(X, Y, W, Z) ;
```

если входной поток имеет вид

X = 5, Y = −6, Z = 7.94, W = 'ABCD';

▲ **3.51.** Указать вид печати при выполнении операторов

1)
```
DECLARE (X, Y, Z) FLOAT;
  X = 10; Y = −8; Z = 0;
  PUT DATA (X, Y, Z);
```

2)
```
DECLARE (X, Y) FIXED (4),
        W CHARACTER (5);
  X = −847; Y = −1000;
  W = 'ABAHC';
  PUT DATA (X, Y, W);
```

▲ **3.52.** Указать вид входного потока, чтобы после выполнения оператора GET DATA переменные имели значения $n = m = 7.8$; $a = 10^{-6}$; $q = -8 \cdot 10^4$; $x = y = z = 1$.

Ввод и вывод значений элементов массивов

Существует несколько способов ввода — вывода массивов.

1. Массив вводится или выводится целиком. В этом случае в операторе ввода — вывода (режим LIST, EDIT, DATA) указывается только имя массива. Элементы массива вводятся — выводятся в порядке их расположения в памяти: значение последнего индекса меняется быстрее остальных. Для двумерных массивов ввод — вывод идет по строкам.

3.53. Определить, какие значения будут присвоены элементам массива A и B после выполнения операторов

```
DECLARE A (10) FIXED (1),
        B (3,4) FIXED (3);
GET EDIT (A, B) (10 F (1), COLUMN (1), 12F (2));
```

если одномерный массив A, состоящий из 10 элементов, отперфорирован на одной перфокарте, двумерный массив B (3×4) — на второй перфокарте. Входной поток имеет вид

```
0 1 2 3 4 5 6 7 8 9 1 0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣..
␣1␣2␣3␣4␣5␣6␣7␣8␣9 1 0 1 1 1 2␣␣␣..
```

2. Индексирование элементов массива (использование оператора цикла DO) [см. 3.5]. Например, операторы

```
DO I=1 T010 BY2;
GET LIST (A(I));
END;
```

или

```
GET LIST ((A(I) DO I=1 TO 10 BY 2));
```

выполняют ввод данных, эквивалентный списку переменных, A(1), A(3), A(5), A(7), A(9).

● **3.54.** Записать операторы ввода, обеспечивающие ввод элементов матрицы **X**(5×4). На перфокартах элементы матрицы отперфорированы по столбцам: на первой — значения элементов первого столбца, на второй — второго и т. д.

В этом случае нельзя воспользоваться предыдущим правилом и указать в списке данных только имя массива, так как размещение элементов массива в памяти не соответствует их расположению на перфокартах. Такой ввод можно выполнить с использованием оператора цикла DO:

```
DECLARE X (5,4) FIXED (7,3);

N: DO J=1 TO 4;

M: DO I=1 TO 5;

     GET LIST (A, (I, J));
     END M;
   END N;
```

Этой группе операторов эквивалентна следующая запись оператора индексированного ввода:

```
DECLARE X (5,4) FIXED (7,3);

GET LIST (((X(I, J)DO I = 1 TO 5) DO J = 1 TO 4))
```

3. С перечислением элементов массива. Этот способ записи списка целесообразно применять в случаях, когда порядок следования значений элементов в списке не соответствует порядку расположения элементов в массиве или когда требуется ввести или вывести значения отдельных элементов массива. Оператор

```
PUT EDIT (Z(5), Z(7))  (SKIP, F(5), X(20), F(5));
```

будет выводить значения пятого и седьмого элементов массива Z.

● **3.55.** Записать операторы ввода, обеспечивающие ввод матрицы A ($m \times n$) ($n \leqslant 10$, $m \leqslant 20$). Значения m и n ввести с отдельной перфокарты. Элементы матрицы определены как FIXED (5, 2) и расположены на перфокартах по столбцам.

Ввод можно описать операторами

```
DECLARE  A (10, 20) FIXED (5,2),
         (M, N) FIXED (2);
/*ВВОД ЗНАЧЕНИЙ MN*/
GET EDIT (M, N)  (COLUMN(1), 2F(2));
/*ВВОД МАССИВА ПО СТОЛБЦАМ*/
GET EDIT (((A, (I, J) DO I=1 TO M) DO J=1 TO N))
         (SKIP, 10 F (5,2));
```

При такой записи осуществляется ввод значений для переменных n и m с одной перфокарты, затем по второму оператору GET в цикле выполняется ввод значений для элементов массива A(1, 1); A(2, 1); A(3, 1); A(4, 1) и т. д. При этом быстрее изменяется первый индекс, так как он включен во внутренний цикл.

3.56. Записать операторы описания и ввода массива Z, состоящего из 250 элементов. Под каждое число отвести пять позиций. Ввод выполнить в режиме LIST.

▲ **3.57.** Записать операторы описания и ввода массивов S и Q, каждый из которых состоит из 200 элементов, используя формат F(6, 2) и располагая на одной перфокарте по 10 чисел.

▲ **3.58.** Записать операторы описания и ввода значений простых переменных x, y, z, отводя под каждое из них по четыре позиции на перфокарте, и полностью массива A, состоящего из 50 элементов. Каждый элемент массива занимает семь позиций, из них три позиции отводятся под дробную часть. На одной перфокарте расположено по пять элементов массива.

3.59. Определить, какие значения будут присвоены элементам массива после выполнения операторов

```
DECLARE A (8) FIXED (3);
GET DATA (A);
```

если входной поток имеет вид

A (1) = 13, A (4) = −8, A (2) = −4 __ A (6) = 1;

▲ **3.60.** Записать операторы описания и ввода в массив, состоящий из 80 элементов, первых k элементов. Значение k ввести с отдельной перфокарты. Элементы массива FIXED(6) расположены по восемь чисел на одной перфокарте.

▲ **3.61.** Определить вид печати при выполнении операторов:

```
DECLARE A (10);
        A (1) = 8; A (2) = 8;
N:        DO I = 3 TO 10;
        A (I) = I * 2 − 10;
        END N;
        PUT EDIT (A) (2(F (3), 2F (5,1), F (6,2)));
```

▲ **3.62.** Записать операторы описания и ввода массивов X, Y, Z, содержащих по 20 элементов каждый, используя формат F(6, 0). На одной перфокарте отперфорировано по одному элементу каждого из массивов, т. е. на первой перфокарте x_1, y_1, z_1, на второй — x_2, y_2, z_2 и т. д.

▲ **3.63.** Записать операторы описания и ввода элементов массивов B, C и D, имеющих описатели

```
DECLARE (B (100), C (4,25), D (10, 5, 2)) FIXED (7,3);
```

если: 1) отперфорированные массивы образуют три колоды перфокарт. На каждой перфокарте отперфорировано по одному элементу массива; 2) то же, что в п. 1, но на каждой перфокарте отперфорировано по пять элементов массива, разделенных пропуском одной колонки; 3) элементы массивов отперфорированы сплошным потоком без пропусков начиная с первой колонки; 4) элементы масси-

вов образуют две колоды перфокарт: в одной колоде — на первой перфокарте — b_1, c_{11}, на второй — b_2, c_{12} и т. д.; во второй колоде — элементы массива D по пять чисел на перфокарте.

3.64. Записать оператор вывода для печати массива Z, состоящего из 100 элементов. В каждой строке отпечатать по 10 чисел по формату F(8, 2), организуя пробелы между числами.

▲ **3.65.** Записать оператор вывода на печать массивов X и Y, состоящих из 25 элементов каждый. Сначала отпечатать элементы массива X, затем массива Y, используя формат E(10, 3). Вывод сопровождать заголовками

МАССИВ X

. . . . . выводимые значения

МАССИВ Y

. . . . . выводимые значения

▲ **3.66.** Записать операторы вывода на печать значений аргумента и функции в двух столбцах, используя формат F (5, 2) и E(12, 3), со следующими заголовками:

АРГУМЕНТ ФУНКЦИЯ

. . . . . } выводимые значения,

если значения аргумента функции сведены в массив X, а вычисленные значения функции — в массив F. Оба массива состоят из 20 элементов.

▲ **3.67.** Записать оператор вывода на печать массивов A, B, C, состоящих из восьми элементов каждый. В строке печатать номер элемента и по одному элементу каждого массива по формату E(20, 5).

3.68. Записать оператор вывода на печать в общепринятом виде матрицы **Z** (5×10), используя формат F (8, 3) и организуя пробелы между числами.

▲ **3.69.** Решить задачу 3.68, выводя на печать слева перед каждой строкой номер строки.

▲ **3.70.** Решить задачу 3.68, выводя на печать номера строк и номера столбцов. Номер столбца напечатать над серединой выводимого значения элемента. Номера столбцов сведены в массив NUMER.

▲ **3.71.*** Записать операторы вывода на печать левой треугольной матрицы, используя формат F(7, 2), если дана квадратная матрица A(10×10).

3.72. Решить задачу 3.71, выводя на печать правую треугольную матрицу.

▲ **3.73.*** Определить, какие значения будут присвоены элементам целочисленного массива A, состоящего из 10 элементов, при выполнении операторов

```
N: DO K = 1 TO 10;
   GET EDIT (A (K)) (SKIP, COLUMN (2 * K), F (1));
   END N;
```

* Ответ к задаче см. на с. 120.

если колода перфокарт состоит из десяти одинаковых карт:

0 1 2 3 4 5 6 7 8 9 ⎵ ⎵ ⎵…

0 1 2 3 4 5 6 7 8 9 ⎵ ⎵ ⎵…

. . . . . .

0 1 2 3 4 5 6 7 8 9 ⎵ ⎵ ⎵…

▲ **3.74.** Записать оператор ввода значений элементов второго и третьего столбцов матрицы A, объявленной оператором

```
DECLARE A (5,5) FIXED (5,1);
```

в режиме LIST.

3.75.* Записать оператор вывода на печать (режим DATA) значений элементов одномерных массивов X, Y, Z, объявленных оператором

```
DECLARE (X, Y, Z) (10) FIXED (3);
```

в следующем порядке: 1) $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, \ldots, x_{10}, y_{10}, z_{10}$; 2) $x_{10}, y_{10}, z_{10}, x_9, y_9, z_9, \ldots, x_1, y_1, z_1$; 3) $x_1, x_2, \ldots, x_{10}, y_1, y_2, \ldots, y_{10}, z_1, z_2, \ldots, z_{10}$; 4) $x_1, y_1, x_2, y_2, x_3, y_3, \ldots, x_{10}, y_{10}, z_1, z_2, \ldots, z_{10}$.

▲ **3.76.*** Определить вид входного потока данных, чтобы после выполнения операторов

```
DECLARE A (0:3, 0:5);
        GET DATA (A);
```

элементам массива A были присвоены значения $a_{00}=a_{01}=10^{-13}$; $a_{12}=a_{34}=a_{23}=18\cdot10^{-15}$. Значения остальных элементов не меняются.

3.77. Указать вид печати после выполнения операторов

```
DECLARE A (5) FIXED (1);
   N: DO I = 1 TO 4;
        A (I) = I + 5;
      END N;
      PUT DATA (A, A (2));
```

3.3. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ЛИНЕЙНОЙ СТРУКТУРЫ

Первым оператором программы должен быть оператор PROCEDURE с описателем OPTIONS (MAIN), указывающий, что выполнение программы начинается с первого выполняемого оператора этой программы (процедуры).

Стандартное начало программы:

имя процедуры: PROCEDURE OPTIONS (MAIN);

Наличие имени процедуры обязательно. Оно не должно превышать семи символов, первый символ — обязательно буква.

Заканчивается программа оператором конца процедуры

```
END;
```

или

```
END⎵ имя процедуры;
```

---

* Ответ к задаче см. на с. 120.

Все переменные, используемые в программе, должны быть описаны либо оператором DECLARE, либо приниматься по соглашению.

Операторы описания данных записываются в начале программы, затем располагаются выполняемые операторы в последовательности, определяемой схемой алгоритма.

Запись операторов в строке бланка производится в позициях с 2-й по 72-ю, причем начало и конец записи могут находиться в произвольном месте строки.

Для лучшей наглядности записей программы, отделения частей программы и описания их содержания используется комментарий вида

/*СТРОКА СИМВОЛОВ*/

Переменные и массивы должны быть определены (заданы своими значениями) до их использования в программе. Для этого применяют либо операторы ввода, либо операторы присваивания.

Оператор присваивания служит для предписания вычисления значения выражения и запоминания этого значения под именем переменной.

Например, Z=A*3.1415—FI; X=X+0.1;

Формат оператора

[МЕТКА:] $v=a$;

где v — идентификатор переменной, значение которой вычисляется; a — арифметическое или логическое выражения.

Метка — это идентификатор, который ставится перед оператором для ссылки на него в тексте программы. Метка отделяется от оператора двоеточием. Идентификатор задается программистом по правилам образования имен.

В зависимости от вида выражения переменная v может быть арифметическая (арифметический оператор присваивания) или логическая (логический оператор присваивания).

В языке ПЛ/1 допускаются многократные присваивания: A, B, C=1+X*3; Эта запись эквивалентна последовательности операторов, т. е. A=1+X**3; B=1+X*3; C=1+X*3;

Допустима конструкция оператора присваивания и для массивов:

A=B; Если A и B — массивы одинаковой размерности, то происходит поэлементное присваивание, т. е. A(1)=B(1); A(2)=B(2) и т. д.

A=X; если A — массив, X — выражение, то каждому элементу массива A присваивается значение X, т. е. A(1)=X; A(2)=X; A(3)=X и т. д.

▲ **3.78** *. Записать операторы присваивания для выражений:

1) $y=\sin^2 a+\cos(a-\pi)+1$; 2) $r=\dfrac{x_i^2+x_{2i}-c}{x_i-b}+b$,

где x_i, x_{2i} — элементы массива;

3) $y=0{,}693147+\dfrac{a}{2\cdot 4+\dfrac{a^3}{(4+a)^3}+\dfrac{a^5}{5(4+a)^5}}$; 4) $w=a^2+1\geqslant 0{,}7$;

5) $q=a>0\wedge b>0$.

● **3.79.** Составить программу решения задачи 1.1.

Схема алгоритма решения задачи приведена на рис. 1.1 и определяет последовательность выполняемых действий. Операторы программы необходимо записывать в том же порядке, что и блоки в схеме алгоритма. Процесс составления программы сводится к представлению блоков алгоритма операторами языка ПЛ/1. Программа имеет вид

---

* Ответ к задаче см. на с. 120.

```
PROG:  PROCEDURE OPTIONS (MAIN);
/*ОПИСАНИЕ ДАННЫХ*/
       DECLARE (A, B, C) FIXED (6,2);
/*ВВОД ДАННЫХ*/
       GET EDIT (A, B, C) (3F(8));
/*ВЫЧИСЛЕНИЯ*/
       P=(A+B+C)/2; T=2*SQRT(P*(P—A)*(P—B)*(P—C));
       HA=T/A; HB=T/B; HC=T/C;
/*ПЕЧАТЬ РЕЗУЛЬТАТОВ*/
       PUT EDIT (HA, HB, HC) (3(F(6,2),X(10)));
/*КОНЕЦ ПРОГРАММЫ*/
     END PROG;
```

3.80. Составить программу решения задачи 1.2. Ввод исходных данных осуществить с перфокарт, расположив на каждой перфокарте по одному числу в формате F (4, 2). Печать результатов выполнить, используя формат F (12, 3).

▲ **3.81.** Составить программу решения задачи 1.3. Значения x_i, y_i, m_i хранятся в массивах. Ввод выполнить с перфокарт, расположив на каждой перфокарте значения x_i, y_i, m_i, относящиеся к одной точке. Для ввода и печати результатов использовать формат F (5,2).

3.82. Составить программу решения задачи 1.4. Все переменные в программе простые.

3.83. Составить программу решения задачи 1.5. Ввод данных выполнить по формату F (4.1), расположив все данные на одной перфокарте. Печать результатов выполнить в формате F (4,2) с заголовком на отдельной строке МЕДИАНЫ ТРЕУГОЛЬНИКА.

3.84. Составить программу решения задачи 1.6. Исходные данные — пятизначные десятичные числа, содержащие две дробные цифры. Печать результатов выполнить по формату E (10, 3).

3.85. Составить программу решения задачи 1.7. Исходные данные — шестизначные десятичные цифры. Печать результатов выполнить по формату F (10, 2).

▲ **3.86.** Составить программу решения задачи 1.8. Исходные данные — пятизначные числа с двумя дробными цифрами. Каждое число отперфорировано на отдельной перфокарте. Печать результатов выполнить по формату F (7, 2) с заголовком ВЫСОТА=.

3.4. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Выполнение операторов в программе осуществляется последовательно в том порядке, как они записаны в программе. Часто в практических расчетах требуется изменять естественный порядок выполнения операторов, организовывать сравнения и в зависимости от результата выполнять те или иные действия. Для этих целей используются операторы управления.

Оператор безусловного перехода GO TO *n* используется для перехода к оператору, помеченному меткой *n*. Например,

```
      GO TO M1;
M2:   F=A*3;
. . . . . .
M1:   F=A*2;
```

Оператор, следующий за оператором GO TO, должен обязательно иметь метку, так как иначе он не будет выполняться.

Условный оператор IF используется для организации ветвлений в программе при выполнении некоторого условия.

Формат условного оператора

[метка:] IF < логическое выражение > THEN < оператор 1 >;

ELSE < оператор 2 >;

Выполнение условного оператора IF заключается в следующем. Сначала вычисляется значение логического выражения; если условие удовлетворяется, то выполняется оператор 1, если нет, то оператор 2. В обоих случаях после выполнения оператора IF выполняется оператор, записанный сразу за ним. Например,

```
IF X>=Y THEN Z=EXP (X);
       ELSE Z=X**3;
PUT DATA (Z);
```

При выполнении оператора проверяется условие X≥Y. Если X≥Y (результат операции '1' B), то выполняется THEN-ветвь, а именно Z=EXP(X), и далее печать Z. В противном случае (результат сравнения '0'B), выполняется ELSE-ветвь, т. е. Z=X**3, и далее печать Z.

Допускаются две модификации оператора IF:

а)

IF < логическое выражение > THEN < оператор 1>;

б)

IF < логическое выражение > THEN; ELSE < оператор 2 >;

В первом случае при истинном значении результата выполняется THEN-ветвь и далее следующий оператор программы. В противном случае — сразу следующий оператор программы (рис. 3.1,*а*). Вторая модификация выполняет противоположные действия (рис. 3.1, *б*).

В операторе IF после слов THEN и ELSE можно поместить только один оператор. Если необходимо разместить группу операторов (рис. 3.1, *в*), то они должны быть объединены в DO-группу с помощью оператора DO:

THEN DO; оператор 1; оператор 2;...; оператор *n*; END;

или

ELSE DO; оператор 1; оператор 2;...; оператор *n*; END;

Например:

```
IF. X<1.5  THEN DO;
                X=A; Z=A**3; END;
           ELSE Z=X;
```

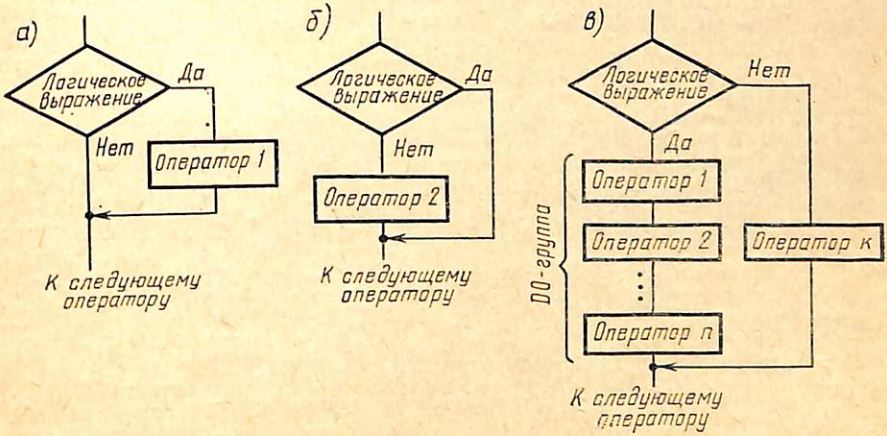


Рис. 3.1

Здесь при X<1,5 выполняется группа операторов X=A и Z=**3, а при X≥≥1,5 — ELSE-ветвь.

В THEN- и ELSE-ветвях могут находиться свои условные операторы IF, в этом случае образуются вложенные операторы IF.

3.87. Составить программу вычисления величины x:

$$x=\begin{cases}(a+b)/5+a, & \text{если } a+b<0;\\ 8,\ 35, & \text{если } a+b\geqslant 0;\end{cases}$$

при $a=-9{,}88$, $b=3{,}41$.

Программа имеет вид

```
FU: PROCEDURE OPTIONS (MAIN);
    A = 9.88; B = 3.41;
    IF A + B < 0 THEN X = (A + B)/5 + A;
                 ELSE X = 8.35;
    PUT DATA (X);
    END FU;
```

Переменные a, b, x по умолчанию принимаются FLOAT(6). Если $a+b<0$, то выполняется оператор присваивания X=(A+B)/5+A. После этого — печать результата. Если $a+b\geqslant 0$, то выполняется оператор X=8.35 и далее печать результата.

● **3.88.** Составить программу решения задачи 1.9.

Схема алгоритма, реализующего этот вычислительный процесс, представлена на рис. 1.2. Для организации разветвлений в программе используем условный оператор IF, проверяющий условие $y=0$. Так как в ветвь ELSE входят блоки *5* и *6*, то для их записи в программе используем DO-группу.

Если $y=0$, то в THEN-ветвь включим оператор печати Y=0,

если $y<0$ или $y>0$, то в ELSE-ветви запишем группу DO, осуществляющую вычисление значения $z=x^3/y$ и его печать.

Программа имеет вид

```
AB:  PROCEDURE OPTIONS (MAIN);
     /*ОПИСАНИЕ ПЕРЕМЕННЫХ*/
     DECLARE (X, Y, Z) FIXED (7,2),
             N FIXED (2);
     /*ВВОД ДАННЫХ*/
     GET DATA (X, N);
     /*ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ Y*/
     Y=SIN (N*X)+0,5;
     /*ПРОВЕРКА УСЛОВИЯ Y=0*/
     IF Y=0 THEN PUT LIST ('Y=0');
            ELSE DO;
                 Z=X**3/Y;
                 PUT DATA (Z);
                 END;
     END AB;
```

● **3.89.** Составить программу решения задачи 1.10.

В соответствии со схемой алгоритма (см. рис. 1.3) программа имеет вид

```
X1: PROCEDURE OPTIONS (MAIN);
```

```
GET EDIT (A, B, X) (3F (6,3));
IF X <= A THEN Z = SIN (X);
        ELSE IF X >= B THEN Z = TAN (X);
                       ELSE Z = COS (X);
PUT DATA (Z);
END X1;
```

Здесь в первом операторе IF в ELSE-ветвь входит второй оператор IF. Если $x \leqslant a$, то вычисляется Z=SIN (X) и далее управление передается оператору, следующему за первым оператором IF, т. е. оператору печати PUT. Если $x > a$, то управление передается второму оператору IF. В зависимости от результата $x \geqslant b$ или $x < b$ выполняется либо Z=TAN (X) и печать z, либо Z=COS (X) и печать z.

▲ **3.90.** Составить программу решения задачи 1.11. Ввод данных a, b, c выполнить в режиме DATA, вывод — в режиме EDIT, сопровождая заголовками КОРНИ ДЕЙСТВИТЕЛЬНЫЕ или КОРНИ МНИМЫЕ.

3.91. Составить программу решения задачи 1.12, используя ввод — вывод в режиме DATA.

▲ **3.92.** Составить программу решения задачи 1.13 для вариантов а), б), в), используя ввод — вывод в режиме EDIT.

▲ **3.93.** Составить программу вычисления значения переменной

$$c = \begin{cases} 1 \text{ В, если } x > 0; \\ 0 \text{ В, если } x \leqslant 0; \end{cases}$$

где x — переменная вещественного типа.

▲ **3.94.** Составить программу решения задачи 1.15. Ввод — вывод выполнить в режиме LIST.

▲ **3.95.** Составить программу решения задачи 1.16. Значение x_0, y_c отперфорировать на одной перфокарте, значение r — на другой. Ввод — вывод выполнить в режиме EDIT.

▲ **3.96.** Составить программу решения задачи 1.17. Ввод — вывод выполнить в режиме DATA.

▲ **3.97.** Составить программу решения задачи 1.18. Ввод — вывод выполнить а) в режиме DATA; б) в режиме EDIT с заголовком.

▲ **3.98.** Составить программу решения задачи 1.19, учитывая, что целая часть от $x+0{,}5$ является целым числом. Ввод — вывод выполнить в режиме DATA.

3.99. Составить программу решения задачи 1.20. Для определения кратности числа использовать функцию MOD.

▲ **3.100.** Составить программу решения задачи 1.20. Для определения кратности числа использовать аксиому: целое число кратно трем, если результат умножения на 3 целочисленного частного от деления x на 3 равен исходному числу.

3.5. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Условия организации циклических участков программы см. в 1.3.

● **3.101.** Составить программу решения задачи 1.21. Схема алгоритма решения задачи представлена на рис. 1.5.

Программа имеет вид

```
TAB:  PROCEDURE OPTIONS (MAIN);
      /*ОПИСАНИЕ И ВВОД ДАННЫХ*/
        DECLARE (X, A) FIXED (5,2);
        GET LIST (A);
      /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ ПАРАМЕТРА ЦИКЛА*/
        X=0;
      /*ВЫЧИСЛЕНИЕ ТЕКУЩЕГО ЗНАЧЕНИЯ Y И ПЕЧАТЬ Y*/
      M1: Y=A**3/(A*A+X*X);
          PUT DATA (Y) SKIP;
      /*ВЫЧИСЛЕНИЕ ТЕКУЩЕГО ЗНАЧЕНИЯ ПАРАМЕТРА ЦИКЛА X*/
          X=X+0.1;
      /*ПРОВЕРКА УСЛОВИЯ НА ПОВТОРЕНИЕ ЦИКЛА*/
          IF X<=3 THEN GO TO M1;
          END TAB;
```

В этой программе в операторе IF ELSE-ветвь опущена, так как содержит оператор END, совпадающий с последним оператором программы.

● **3.102.** Составить программу решения задачи 1.22. Схема алгоритма приведена на рис. 1.7.

Программа имеет вид

```
R:    PROCEDURE OPTIONS (MAIN);
      DECLARE (X, Y, EPS) FLOAT (4);
      GET LIST (X, EPS);
      Y=1; N=1;
T1;   Y=Y*X/N;
      PUT EDIT (Y) (SKIP, F (8,2));
      N=N+1;
      IF Y>EPS THEN GO TO T1;
      END R;
```

Данная программа иллюстрирует итерационный цикл, т. е. цикл, число повторений которого заранее не известно.

Однако в языке ПЛ/1 для организации итерационных циклов можно использовать оператор цикла типа DO WHILE, который строится по следующей схеме:

```
[метка:] DO WHILE (выражение);
           оператор 1;
           оператор 2;   } тело цикла
           оператор n;
         END [метка];
```

Последовательность операторов, образованная опеартором DO, будет выполняться до тех пор, пока значение выражения в заголовке DO истинно. Как только значение выражения станет ложно («нет»), то выполняется оператор программы, следующий за оператором END.

С использованием оператора DO WHILE программа решения задачи 1.22 имеет вид

```
R:  PROCEDURE OPTIONS (MAIN);
    DECLARE (X, Y, EPS) FLOAT (4); GET LIST (X, EPS);
    Y=1; N=1;
```

```
    DO WHILE (Y>EPS);
    Y=Y*X/N;
    PUT EDIT (Y) (SKIP, F(8,2));
    N=N+1;
    END;
  END R;
```

Для организации циклов с заданным числом повторений целесообразно использовать оператор цикла типа DO TO с автоматическим изменением параметра цикла, который строится по схеме

```
[метка:] DO i=m1 TO m2 BY m3;
            оператор 1;
            оператор 2;   } тело цикла
            оператор n;
         END [метка];
```

где i — параметр цикла, в качестве которого обязательно должна быть простая переменная; m_1, m_2, m_3 — выражения, указывающие начальное, конечное значения параметра цикла и шаг его изменения.

Если шаг изменения параметра цикла m_3 равен 1, то его можно опускать в записи оператора DO. Например,

```
    S=0;
M:  DO I=1 TO 10;
    S=S+A(I);
    END M;
```

Оператор S=S+A(I) выполняется 10 раз, причем значение параметра цикла изменяется от 1 до 10 с шагом 1. После этого управление передается оператору, следующему за END M. Значение i в этот момент равно 11.

Допустимы также конструкции оператора DO с отрицательным и переменным шагом. Например,

```
DO X=10.5 TO 0 BY—0.5;
DO I=1 TO 6 BY2, 50 TO 40 BY—2;
```

Во втором операторе параметр цикла i изменяется от 1 до 6 с шагом 2, а затем от 50 до 40 с шагом —2.

Все указанные выше заголовки оператора DO могут содержать конструкцию WHILE (выражение). Например,

```
DO X=0.1 BY 0.01 WHILE (X<=1);
```

Переменная X будет изменяться от 0,1 с шагом 0,01 до тех пор, пока не станет больше 1.

Входить в цикл можно только через заголовок DO. Начальное, конечное значения и шаг изменения параметра, указанные в заголовке оператора DO, внутри цикла изменять нельзя.

● **3.103.** Составить программу решения задачи 1.21 с использованием оператора цикла типа DO TO.

Программа имеет вид

```
TAB: PROCEDURE OPTIONS (MAIN);
     DECLARE (X, A) FIXED (5,1),
             Y FIXED (10, 2); GET LIST (A);
        DO X=0 BY 0.1 TO 3;
        Y=A**3/(A*A+X*X);
        PUT EDIT (Y) (SKIP, F (10, 2));
        END;
     END TAB;
```

Здесь x — параметр цикла, его начальное значение принимается равным 0. Сначала вычисляется значение y и производится печать. Затем параметр цикла автоматически увеличивается на шаг, рав-

ный 0,1, и процесс вычисления y и печати его значения повторяется. Так будет до тех пор, пока x не станет больше 3. Происходит выход из цикла и выполняется следующий за END оператор END TAB.

● **3.104.** Составить программу решения задачи 1.23, используя оператор цикла.

Программа имеет вид

```
W1:  PROCEDURE OPTIONS (MAIN);
     DECLARE (X, A) (40) FIXED (6,2);
     GET LIST (A, X);
     /*ЦИКЛ НА 40 ПОВТОРЕНИЙ*/
R:   DO I=1 TO 40;
          Z=SQRT ((X(I)+A(I))/2);
          PUT EDIT (Z) (COLUMN (1), F(10, 2));
          END R;
     END W1;
```

Исходные массивы A и X описаны в операторе DECLARE. Оператор GET осуществляет ввод данных, управляемый списком. Оператор DO организует цикл с параметром i, который изменяется от 1 до 40. Внутри цикла выполняется оператор присваивания, вычисляющий значение z, и печати. Вид печати — печать столбцом, в строке — одно число.

3.105. Составить программу решения задачи 1.24, используя оператор цикла.

▲ **3.106.** Составить программу решения задачи 1.25, используя оператор цикла.

▲ **3.107.** Составить программу решения задачи 1.26, используя оператор цикла.

▲ **3.108.** Составить программу решения задачи 1.27, используя для организации цикла а) условный оператор IF; б) оператор цикла DO WHILE.

▲ **3.109.** Составить программу решения задачи 1.28, используя условный оператор IF.

3.110. Составить программу решения задачи 1.29, используя оператор цикла DO TO.

▲ **3.111.** Составить программу решения задачи 1.30, используя а) условный оператор внутри цикла, б) оператор цикла DO WHILE.

▲ **3.112.** Составить программу решения задачи 1.31, используя оператор цикла типа DO TO.

3.113. Составить программу решения задачи 1.32, используя оператор цикла DO TO. На печать выдать все значения a_i, b_i, c_i и z_i.

▲ **3.114.** Составить программу решения задачи 1.33, используя условный оператор IF.

▲ **3.115.** Составить программу решения задачи 1.34, используя оператор цикла типа DO TO. Исходные числа заданы на перфокартах парами x_i, y_i. На каждой перфокарте по пять пар чисел.

▲ **3.116.** Составить программу решения задачи 1.35, используя условный оператор IF.

3.117. Составить программу решения задачи 1.36, используя а) условный оператор IF; б) оператор цикла DO WHILE.

▲ **3.118.** Составить программу решения задачи 1.37, используя а) условный оператор IF; б) оператор цикла DO TO.

▲ **3.119.** Составить программу решения задачи 1.38, используя оператор цикла DO TO, считая $n \leqslant 100$.

3.120. Составить программу решения задачи 1.39, используя а) условный оператор IF, б) оператор цикла DO WHILE. Цикл должен заканчиваться, когда значение y станет меньше или равным нулю.

▲ **3.121.** Составить программу решения задачи 1.40, используя оператор цикла DO TO и обозначения $c=\cos(n-1)x$; $a=\cos x$; $b=\sin x=\sqrt{1-\cos^2 x}$; $d=\sin(n-1)x$.

▲ **3.122.** Составить программу решения задачи 1.41, используя условный оператор IF.

▲ **3.123.** Составить программу решения задачи 1.42. Условие кратности числа трем см. в задаче 3.100.

3.6. ХАРАКТЕРНЫЕ ПРИЕМЫ ПРОГРАММИРОВАНИЯ

Вычисление в цикле с несколькими одновременно изменяющимися параметрами

Оператор цикла языка ПЛ/1 позволяет задавать закон изменения только одного параметра. Для задания закона изменения всех остальных параметров необходимо использовать операторы присваивания, которые перед началом цикла устанавливают начальные значения, а в цикле вычисляют их текущие значения.

● **3.124.** Составить программу решения задачи 1.43. Параметр цикла i изменять с помощью оператора цикла типа DO TO, а значение x — с помощью оператора присваивания.

Программа, реализующая схему алгоритма (см. рис. 1.8), имеет вид

```
AC:    PROCEDURE OPTIONS (MAIN);
       DECLARE Y (20) FIXED (5,2),
                 (A, H, Z) FIXED (8,2);
       GET LIST (Y, A, H);
       /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ X*/
       X=A;
       /*ЦИКЛ ПО ПАРАМЕТРУ I*/
CIKL:  DO I=1 TO 20;
       Z=X*Y(I)/(X+Y(I));
       /*ВЫЧИСЛЕНИЕ ТЕКУЩЕГО ЗНАЧЕНИЯ X*/
       X=X+H;
       PUT EDIT (Z) (SKIP, F(8,2));
       END CIKL;
END    AC;
```

3.125. Составить программу решения задачи 1.44. Для ввода данных и печати результата использовать режим LIST.

▲ **3.126.** Составить программу решения задачи 1.45. На печать выводить значения x, a_i, b_i, c_i и соответствующее значение z. Режим печати EDIT.

3.127. Составить программу решения задачи 1.46. На печать выдавать значение i и значение функции u.

▲ **3.128.** Составить программу решения задачи 1.47. Для организации цикла по переменной x использовать оператор DO TO.

▲ **3.129.** Составить программу решения задачи 1.48. Значения x и h ввести с перфокарт. Для организации цикла по переменной n использовать оператор типа DO TO.

▲ **3.130.** Составить программу решения задачи 1.49. Значения x_0, n_0, m, Δx, Δn ввести с перфокарт. Для организации цикла по переменной n использовать оператор DO TO.

Запоминание результатов

● **3.131.** Составить программу решения задачи 1.50.

Для запоминания результатов необходимо выделить массив, который должен быть описан в операторе DECLARE. В программе результат вычисления z_i определять как переменную с индексом. Программа, реализующая схему алгоритма (см. рис. 1.9), имеет вид

```
M:  PROCEDURE OPTIONS (MAIN);
    /*ОПИСАНИЕ ВХОДНОГО И РЕЗУЛЬТИРУЮЩЕГО МАССИВА*/
    DECLARE (X, Z) (50);
    /*ВВОД МАССИВА X*/
    GET LIST (X);
    /*ЦИКЛ ПО ПАРАМЕТРУ I ДЛЯ ВЫЧИСЛЕНИЯ Z (I)*/
A:  DO I=1 TO 50;
        Z(I)=SQRT((X(I)**2+1)/I);
        END A;
    PUT EDIT (Z) (SKIP, 10(F(8,2), X(4)));
    END M;
```

3.132. Составить программу решения задачи 1.51, используя приемы запоминания результатов и организации цикла с несколькими одновременно изменяющимися параметрами.

3.133. Составить программу решения задачи 1.52, используя для организации цикла по переменной i оператор DO TO. Вывод организовать, печатая по пять элементов в строке.

▲ **3.134.** Составить программу решения задачи 1.53, используя для организации цикла по переменной i оператор DO TO. Для печати значений массива Y использовать режим DATA.

3.135. Составить программу решения задачи 1.54, учитывая, что $y_k = x_{41-k}$. На печать выводить массивы X и Y с соответствующими заголовками.

▲ **3.136.** Составить программу решения задачи 1.55. Чтобы элементы записывались в массив Y подряд, необходимо изменять индекс переменной массива Y всякий раз, когда осуществляется запись положительного элемента в массив Y. Вывести на печать массив Y.

3.137. Составить программу решения задачи 1.56 (см. пояснения к задаче 3.136).

▲ **3.138.** Составить программу решения задачи 1.57, для организации цикла используя а) оператор DO TO и условный оператор, проверяющий, что в массив Y записано 10 элементов; б) оператор типа DO TO WHILE.

▲ **3.139.** Составить программу решения задачи 1.58, используя приемы запоминания результатов и организации цикла с несколькими одновременно изменяющимися параметрами, полагая размер массива Z не более 50 элементов.

3.140. Составить программу решения задачи 1.59, учитывая, что индексы элементов массивов A и B принимают разные значения. Значение индекса для элементов массива A определять как $i*2$, где $i=1\div37$.

▲ **3.141.** Составить программу решения задачи 1.60, учитывая, что индексы элементов массивов A, B, C принимают разные значения.

▲ **3.142.** Составить программу решения задачи 1.61, используя для организации цикла по переменной x оператор DO TO. Как только функция y станет отрицательной, выйти из цикла.

▲ **3.143.** Составить программу решения задачи 1.62, учитывая, что индексы элементов массивов X, Y, Z принимают различные значения.

3.144. Составить программу решения задачи 1.63, используя для организации цикла по переменной n оператор DO TO. Учесть что значения индексов элементов массивов A и B не совпадают со значением переменной n.

3.145. Составить программу решения задачи 1.64. Элемент главной диагонали матрицы **A** — переменная с двумя одинаковыми индексами a_{ii}. Элемент одномерного массива B — переменная с одним индексом. Для организации цикла использовать оператор DO TO.

Вычисление суммы и произведения

3.146. Составить программу решения задачи 1.65.

Для накопления суммы необходимо: а) задать перед началом цикла начальное значение суммы $z=0$; б) внутри цикла накапливать сумму, используя зависимость $z=z+x_i^2/i$, где x_i^2/i — текущее значение слагаемого.

Программа имеет вид

```
SUM:  PROCEDURE OPTIONS (MAIN);
      DECLARE X (20);
      GET LIST (X);
      /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ СУММЫ*/
      Z=0;
      /*НАКАПЛИВАНИЕ СУММЫ*/
      DO I=1 TO 20;
          Z=Z+X(I)**2/I;
          END;
      /*ПЕЧАТЬ СУММЫ*/
       PUT EDIT (Z) (E (12,3));
    END SUM;
```

● **3.147.** Составить программу решения задачи 1.66.

Для вычисления произведения необходимо: а) задать перед началом цикла начальное значение $z=1$; б) внутри цикла вычислять произведение, используя зависимость $z=zx_i$, где x_i — текущее значение положительного сомножителя.

Программа имеет вид

```
PROIS:  PROCEDURE OPTIONS (MAIN);
        DECLARE X (100);
        GET EDIT (X) (SKIP, 10 F(5,2));
             Z=1;
             DO I=1 TO 100;
             IF X (I)>0 THEN Z=Z*X(I); END;
        PUT EDIT ('ПРОИЗВЕДЕНИЕ=', Z) (A, E (10,3));
        END PROIS;
```

3.148. Составить программу решения задачи 1.67, используя приемы накопления суммы и организации цикла с несколькими одновременно изменяющимися параметрами.

3.149. Составить программу решения задачи 1.68.

3.150. Составить программу решения задачи 1.69.

3.151. Составить программу решения задачи 1.70.

3.152. Составить программу решения задачи 1.71. Параметр цикла i изменять от 2 до 78 с шагом, равным 2.

▲ **3.153.** Составить программу решения задачи 1.72. Начальное значение суммы взять равным 1, так как нет смысла вычислять значение первого слагаемого. Вычисление каждого текущего значения слагаемого производить по предыдущему значению.

▲ **3.154.** Составить программу решения задачи 1.73. Вычислять сумму и количество отрицательных элементов массива.

▲ **3.155.** Составить программу решения задачи 1.74. Вычислять сумму и количество элементов, лежащих в диапазоне чисел $1 \leqslant a_i \leqslant 2$. Если таких элементов нет, вывести на печать текстовое сообщение.

3.156. Составить программу решения задачи 1.75,

▲ **3.157.** Составить программу решения задачи 1.76.

3.158. Составить программу решения задачи 1.77. Начальное значение суммы ряда взять равным единице. Текущее значение слагаемого ряда вычислять, используя зависимость $y = yx^2/[2n(2n-1)]$.

3.159. Составить программу решения задачи 1.78, используя приемы накопления суммы и произведения.

3.160. Составить программу решения задачи 1.79, используя прием накопления произведения.

3.161. Составить программу решения задачи 1.80. Элементы главной диагонали имеют одинаковые индексы, т. е. a_{ii}. Использовать метод накопления суммы в цикле с параметром i. Для организации цикла использовать оператор DO TO.

3.162. Составить программу решения задачи 1.81.

3.163. Составить программу решения задачи 1.82. В цикле вычислять значения функции z и в зависимости от знака накапливать соответствующие значения сумм.

▲ **3.164.** Составить программу решения задачи 1.83. В цикле вычислять произведение и количество элементов, больших значения a.

▲ **3.165.** Составить программу решения задачи 1.84. В цикле вычислять произведение и количество положительных элементов. Параметр цикла изменять от 2 с шагом 2. Если положительных элементов в массиве нет, вывести на печать текстовое сообщение и значение среднего геометрического не вычислять.

▲ **3.166.** Составить программу решения задачи 1.85. Вычисление текущего значения слагаемого производить, используя зависимость $y=-yx^2/[k(k+n)4]$.

3.167. Составить программу решения задачи 1.86, используя метод накопления произведения в цикле с параметром цикла, изменяющимся от 1 до $m-1$. Начальное значение произведения принять равным n.

3.168. Составить программу решения задачи 1.87. Условие кратности числа трем см. в задаче 3.100.

▲ **3.169.** Составить программу решения задачи 1.88, учитывая, что число кратно двум, если результат умножения самого числа на целочисленный результат от деления числа на два есть само число.

▲ **3.170.** Составить программу решения задачи 1.89, учитывая, что ближайшим целым числа x_i является целая часть от числа $x_i+0{,}5$.

▲ **3.171.** Составить программу решения задачи 1.90.

Вычисление суммы членов бесконечного ряда

▲ **3.172.** Составить программу решения задачи 1.91. Цикл организовать с помощью условного оператора.

Программа, реализующая схему алгоритма, показанную на рис. 1.12, имеет вид

```
R1: PROCEDURE OPTIONS (MAIN);
    GET DATA (X, EPS);
    /*ЗАДАНИЕ НАЧАЛЬНЫХ ЗНАЧЕНИЙ ЧЛЕНА РЯДА, СУММЫ
    И НОМЕРА ЧЛЕНА РЯДА*/
    Y=1; Z=1; N=1;
    /*ВЫЧИСЛЕНИЕ ТЕКУЩЕГО ЗНАЧЕНИЯ Y*/
M1: Y=Y*(-X*X)/((2*N-1)*2*N);
    /*НАКОПЛЕНИЕ СУММЫ*/
    Z=Z+Y;
    /*ИЗМЕНЕНИЕ НОМЕРА ЧЛЕНА РЯДА*/
    N=N+1;
    /*ПРОВЕРКА УСЛОВИЯ НА ПРОДОЛЖЕНИЕ ЦИКЛА*/
    IF Y>EPS THEN GO TO M1;
    PUT DATA (Z);
    END R1;
```

Точность EPS, как правило, задается в виде константы с порядком. Поэтому по соглашению переменные EPS, X задаются с плавающей точкой.

Для организации цикла в подобных задачах удобнее воспользоваться оператором DO WHILE.

Программа имеет вид

```
R1:   PROCEDURE OPTIONS (MAIN);
      GET DATA (X, EPS);
      /*ЗАДАНИЕ НАЧАЛЬНЫХ ЗНАЧЕНИЙ*/
      Z, Y, N=1;
      /*ВЫПОЛНИТЬ ЦИКЛИЧЕСКИ ВСЮ ГРУППУ ОПЕРАТОРОВ,
          ПОКА Y>EPS*/
CIKL: DO WHILE (Y>EPS);
          Y=Y*(-X*X)/((2*N-1)*2*N);
          Z=Z+Y; N=N+1;
          END CIKL;
```

```
        PUT DATA (Z);
        END R1;
```

3.173. Составить программу решения задачи 1.92, используя для организации цикла оператор DO WHILE.

3.174. Составить программу решения задачи 1.93, используя для вычисления текущего значения члена ряда рекуррентную формулу, для организации цикла — оператор DO WHILE.

3.175. Составить программу решения задачи 1.94, используя для организации цикла а) оператор DO WHILE; б) условный оператор IF.

▲ **3.176.** Составить программу решения задачи 1.95, используя для организации цикла условный оператор IF.

3.177. Составить программу решения задачи 1.96, используя для вычисления текущего значения члена ряда рекуррентную формулу, для организации цикла — оператор DO WHILE.

▲ **3.178.** Составить программу решения задачи 1.97, используя для организации цикла оператор DO WHILE.

▲ **3.179.** Составить программу решения задачи 1.98, используя для вычисления текущего значения члена ряда рекуррентную формулу, для организации цикла — условный оператор IF.

Вычисление полинома

● **3.180.** Составить программу решения задачи 1.99.

Программа, реализующая схему алгоритма, показанную на рис. 1.13 при $n=8$, имеет вид

```
POLI:  PROCEDURE OPTIONS (MAIN);
       DECLARE (X, A (9) FIXED (4, 0);
       GET LIST (X, A);
       /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ ПОЛИНОМА*/
       Y=A(1);
       /*ЦИКЛ ПО ПАРАМЕТРУ I*/
CIKL:  DO I=2 TO 9;
       /*ВЫЧИСЛЕНИЕ ТЕКУЩЕГО ЗНАЧЕНИЯ ПОЛИНОМА*/
            Y=Y*X+A(I);
            END CIKL;
       PUT DATA (Y);
       END POLI;
```

Программа для вычисления полинома любой степени n ($n \leqslant 50$) имеет вид

```
POLI:  PROCEDURE OPTIONS (MAIN);
       DECLARE (X, A (51)) FIXED (4,0);
       GET LIST (N, X, (A(I)DOI=1 TO N+1));
       Y=A(1);
CIKL:  DO I=2 TO N+1;
            Y=Y*X+A(I);
            END CIKL;
       PUT DATA (Y);
       END POLI;
```

3.181. Составить программу решения задачи 1.100.

3.182. Составить программу решения задачи 1.101.

▲ **3.183.** Составить программу решения задачи 1.102.
▲ **3.184.** Составить программу решения задачи 1.103.
▲ **3.185.** Составить программу решения задачи 1.104.

Нахождение наибольшего и наименьшего

● **3.186.** Составить программу решения задачи 1.105.

Программа, реализующая схему алгоритма, представленную на рис. 1.14, имеет вид

```
MIN:  PROCEDURE OPTIONS (MAIN);
      GET LITS (A, B, C, H, OMEGA, FI);
      /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ НАИМЕНЬШЕГО*/
      YMIN=1E19;
      /*ЦИКЛ ПО ПАРАМЕТРУ X*/
CIKL: DO X=0 TO C BY H;
         Y=A*EXP(-B*X)*SIN(OMEGA*X+FI);
      /*НАХОЖДЕНИЕ НАИМЕНЬШЕГО ПО СРАВНЕНИЮ С ТЕКУЩИМ*/
         IF Y<YMIN THEN YMIN=Y;
         END CIKL;
      PUT DATA (YMIN);
      END MIN:
```

Условный логический оператор, включенный в цикл, сравнивает вычисленное значение функции y со значением y_{min}. Если $y<y_{min}$, то выполняется присваивание y_{min} значения y. Если $y\geqslant y_{min}$, то идет продолжение цикла.

● **3.187.** Составить программу решения задачи 1.106.

Программа, реализующая схему алгоритма, представленную на рис. 1.15, имеет вид

```
MAX:  PROCEDURE OPTIONS (MAIN);
      DECLARE (XMAX, X(40)) FIXED (8,2);
      GET LIST (X);
      /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ НАИБОЛЬШЕГО*/
      XMAX=X(1);
      /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ НОМЕРА НАИБОЛЬШЕГО*/
      NMAX=1;
      /*ПРОСМОТР ВСЕХ ЭЛЕМЕНТОВ МАССИВА*/
CIKL: DO I=2 TO 40;
         IF X(I)>XMAX THEN DO;
                           XMAX=X(I);
                           NMAX=I; END;
         END CIKL;
      PUT DATA (NMAX, XMAX);
      END MAX;
```

Если $x_i>x_{max}$, то выполняется DO-группа — два оператора присваивания; в противном случае — переход на продолжение цикла.

● **3.188.** Составить программу решения задачи 1.107.

Программа, реализующая схему алгоритма, представленную на рис. 1.17, имеет вид

```
EXTR: PROCEDURE OPTIONS (MAIN);
      /*ВВОД ДАННЫХ*/
      GET LIST (A, B, C, H);
      /*ПРОВЕРКА ЗНАКА C И ВЫЧИСЛЕНИЕ N*/
      IF C>0 THEN N=1; ELSE N=-1;
      /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ ЭКСТРЕМУМА*/
```

```
      YM=N*1E 19;
      /*ОРГАНИЗАЦИЯ ЦИКЛА И ПРОВЕРКА УСЛОВИЯ*/
CIKL: DO X=0 TO 4 BY H;
         Y=ABS (A)*EXP (B*X+C*X*X);
         IF N*Y<N*YM THEN YM=Y; ELSE GO TO R1;
         END CIKL;
R1:   PUT DATA (N, YM);
      END EXTR;
```

В цикле с параметром x осуществляется проверка условия $Ny<Ny_m$. Если это условие истинно, то значение y считается новым экстремальным значением. В противном случае происходит выход из цикла и выполняется печать результата.

3.189. Составить программу решения задачи 1.108.

3.190. Составить программу решения задачи 1.109.

3.191. Составить программу решения задачи 1.110.

▲ **3.192.** Составить программу решения задачи 1.111. Перед циклом задать разные начальные значения, например $y_{max}=-10^{18}$, $y_{min}=10^{18}$ наибольшего и наименьшего значений функции. Поскольку функция сначала достигает максимума, в цикле сначала находить наибольшее, проверяя условие $y>y_{max}$. Как только это условие не выполнится, переходить к отысканию наименьшего, т. е. проверке условия $y<y_{min}$. Если y окажется больше y_{min}, необходимо выходить из цикла на печать результатов.

3.193. Составить программу решения задачи 1.112 (см. пояснения к задаче 3.192).

▲ **3.194.** Составить программу решения задачи 1.113 (см. пояснение к задаче 1.107).

▲ **3.195.** Составить программу решения задачи 1.114. Вычислять порядковые номера максимального и минимального элементов, а потом на их места записать соответствующие значения.

3.196. Составить программу решения задачи 1.115. В цикле находить наибольший и наименьший элементы и их порядковые номера. После цикла поменять местами наибольший элемент и первый и наименьший с последним.

▲ **3.197.** Составить программу решения задачи 1.116.

3.198. Составить программу решения задачи 1.117. Функция сначала достигает максимума. После нахождения максимума надо отыскивать минимум. Как только он будет найден, выходить из цикла (см. пояснение к задаче 3.192).

▲ **3.199.** Составить программу решения задачи 1.118. Так как элементы диагонали a_{ii} имеют два одинаковых индекса, находить наибольший элемент, организуя цикл по переменной i, и определять его порядковый номер.

▲ **3.200.** Составить программу решения задачи 1.119. Внутри цикла проверять, является ли x_i положительным числом и большим по сравнению с наибольшим из просмотренных.

Уточнение корней уравнения

3.201. Составить программу решения задачи 1.120.

Программа, реализующая схему алгоритма, предоставленную на рис. 1.18, имеет вид

```
KORNI:  PROCEDURE OPTIONS (MAIN);
        /*ЗАДАНИЕ НАЧАЛЬНОГО ЗНАЧЕНИЯ КОРНЯ*/
        X 0=4.7;
        /*ВЫЧИСЛЕНИЕ СЛЕДУЮЩЕГО ЗНАЧЕНИЯ КОРНЯ*/
M1:     X1=ATAN (X0)+3.14159;
        /*ПРОВЕРКА ПОЛУЧЕННОЙ ТОЧНОСТИ ВЫЧИСЛЕНИЯ*/
        IF ABS (X1—X0)<=1E—5
                THEN PUT DATA (X0, X1);
                ELSE DO; X0=X1; GO TO M1; END;
        END KORNI;
```

Если при выполнении условного оператора IF установлено, что требуемая точность достигнута, то выполняется печать результата, в противном случае выполняется DO-группа, включающая в себя оператор присваивания нового начального значения корня и оператор безусловного перехода на циклический участок программы.

3.202. Составить программу решения задачи 1.121.

▲ **3.203.** Составить программу решения задачи 1.122.

3.204. Составить программу решения задачи 1.123.

▲ **3.205.** Составить программу решения задачи 1.124.

Программирование вычислительных процессов со структурой вложенных циклов

Структура правильно организованного вложенного цикла может иметь вид

```
A : DO I
. . . . . .
          B : DO J
. . . . . .
          END B;
. . . . . .
          C : DO X
. . . . . .
          END C;
. . . . . .
END A;
```

Каждому оператору цикла DO должен соответствовать свой оператор END. Области действия циклов перекрываться не должны. Обращение к операторам циклов разрешается только через заголовок цикла. Если внутри цикла есть операторы с меткой, то передавать им управление можно только внутри данного цикла.

● **3.206.** Составить программу решения задачи 1.125.

Программа, реализующая схему алгоритма, представленную на рис. 1.19, имеет вид

```
PS:  PROCEDURE OPTIONS (MAIN);
     DECLARE A (10,8);
     /*ВВОД МАТРИЦЫ ПО СТРОКАМ*/
     GET LIST (A);
     /*ВНЕШНИЙ ЦИКЛ — ПЕРЕБОР СТРОК*/
K1:  DO I=1 TO 10;
```

```
      S=0
      /*ВНУТРЕННИЙ ЦИКЛ — ПЕРЕБОР ЭЛЕМЕНТОВ ПО СТРОКЕ*/
         B1 : DO J=1 TO 8;
                  IF A (I, J)>0 THEN S=S+A(I, J);
         END B1;
      PUT DATA (S) SKIP;
      END K1;
END PS;
```

Поскольку сумма накапливается для каждой строки матрицы **A**, оператор **S=0** находится перед внутренним циклом.

● **3.207.** Составить программу решения задачи 1.126.

Программа, реализующая схему алгоритма, представленную на рис. 1.20, имеет вид

```
SUM: PROCEDURE OPTIONS (MAIN);
     DECLARE X(100) FIXED (4,1);
     GET LIST (X);
   A: DO K=1 TO 99;
         XMIN=X(K);
         N=K;
   B: DO J=K+1 TO 100;
         IF X(J)<XMIN THEN DO;
         XMIN=X(J); N=J; END;
         END B;
         X(N)=X(K);
         X(K)=XMIN;
         END A;
     PUT EDIT (X) (SKIP, 10 (F(4,1), X(6)));
     END SUM;
```

3.208. Составить программу решения задачи 1.127. Оператор, задающей начальное значение суммы, равное нулю, должен выполняться один раз и находиться перед внешним циклом, поскольку здесь накапливается одно значение суммы. Оператор печати должен стоять за внешним циклом, так как печать выполняется также один раз после вычисления всей суммы.

▲ **3.209.** Составить программу решения задачи 1.128. Печать результатов выполнить за один раз, поскольку они фиксированы в массиве Z.

▲ **3.210.** Составить программу решения задачи 1.129. Индексы суммируемых элементов изменять следующим образом: номер строки i от 1 до 19, так как в 20-й строке нет элемента, лежащего над главной диагональю; номера столбцов j в i-й строке от $i+1$ до 20.

3.211. Составить программу решения задачи 1.130. В циклах находить наибольшие значения элементов и их порядковые номера и менять местами с i-ми элементами массива.

3.212. Составить программу решения задачи 1.131. На печать вывести весь массив результата.

3.213. Составить программу решения задачи 1.132. На печать вывести вычисленное значение среднего арифметического.

▲ **3.214.** Составить программу решения задачи 1.133. Во внутреннем цикле находить сумму элементов строки, во внешнем — наименьшую из этих сумм и номер ее строки.

3.215. Составить программу решения задачи 1.134. В циклах (внешнем и внутреннем) выполнять присваивание $y_{ji}=x_{ij}$.

▲ 3.216. Составить программу решения задачи 1.135. На печать вывести значения элементов массива C.

▲ 3.217. Составить программу решения задачи 1.136.

▲ 3.218. Составить программу решения задачи 1.137. На печать вывести значения элементов массива M.

3.219. Составить программу решения задачи 1.138. Во внутреннем цикле находить наибольший элемент массива и его порядковый номер. Во внешнем цикле — записывать элемент в текущую ячейку массива Y, а на место наибольшего элемента в массив X записывать очень малое число, например — 10^{10}, чтобы снова этот элемент не был найден как наибольший.

3.220. Составить программу решения задачи 1.139.

▲ 3.221. Составить программу решения задачи 1.140. Сначала организовать два вложенных цикла для нахождения наименьшего элемента матрицы и номера строки и столбца, где он находится. Затем организовать два независимых цикла для записи нулей в найденную строку и столбец. Значения элементов матрицы вывести на печать в общепринятом виде.

▲ 3.222. Составить программу решения задачи 1.141.

3.223. Составить программу решения задачи 1.142. Во внешнем цикле три раза повторить цикл для нахождения наибольшего элемента массива. Вместо наибольшего элемента в исходный массив записать очень малое число, например — 10^{10}.

▲ 3.224. Составить программу решения задачи 1.143. На печать вывести значения элементов результирующей матрицы в общепринятом виде. Считать n и $l\leqslant 10$, $m\leqslant 20$.

▲ 3.225. Составить программу решения задачи 1.144. Организовать внешний цикл на три повторения. В цикле выполнять ввод одного массива, вычисления и печать результатов.

▲ 3.226. Составить программу решения задачи 1.145.

3.227. Составить программу решения задачи 1.146. Организовать внешний цикл на 10 повторений для вычисления значений y_i и вывести их на печать.

● 3.228. Составить программу решения задачи 1.147.

Программа, реализующая схему алгоритма, представленного на рис. 1.21, имеет вид

```
ARG: PROCEDURE OPTIONS (MAIN);
     /*ВВОД ЗНАЧЕНИЯ A*/
     GET DATA (A);
     /*ЗАДАНИЕ НАЧАЛЬНЫХ ЗНАЧЕНИЙ*/
     YMIN=1E 19; X=0.2; H=0.2;
     /*ЦИКЛ ДЛЯ ОПРЕДЕЛЕНИЯ ЗНАЧЕНИЯ YMIN*/
CIKL: DO I=1 TO 51;
          Y=A*X—LOG(X);
          IF Y>=YMIN THEN GO TO M8;
          YMIN=Y; XMIN=X; X=X+H;
          END CIKL;
     /*ВЫЧИСЛЕНИЕ XMIN С ЗАДАННОЙ ТОЧНОСТЬЮ*/
  M8: IF H<=0.01 THEN PUT DATA (XMIN);
```

```
        ELSE DO; X=XMIN—0.2; H=0.01; GO TO CIKL;
        END;
    END ARG;
```

Здесь при первом выполнении циклического участка программы с именем CIKL грубо вычисляется значение аргумента x_{min}. Затем условный оператор проверяет значение шага h. Если требуемая точность не достигнута, то выполняется ELSE-ветвь, содержащая DO-группу: вычисляются значение x, предшествующее x_{min}; новое значение шага и повторяется цикл DO для вычисления значения y_{min} с требуемой точностью. Внешний цикл здесь выполняется дважды, и для его организации используется условный оператор перехода GO TO CIKL.

3.229. Составить программу решения задачи 1.148. Решать задачу, используя методику решения задачи 3.228.

▲ **3.230.** Составить программу решения задачи 1.149. Значения элементов матрицы вывести на печать в общепринятом виде.

▲ **3.231.** Составить программу решения задачи 1.150. Использовать методику решения задачи 3.228.

▲ **3.232.** Составить программу решения задачи 1.151. Организовать два цикла. Во внутреннем цикле сначала находить значение максимума, а потом значение минимума функции. Если минимум пройден, то выполнить выход из внутреннего цикла, записать найденные значения максимума и минимума в массив Z и повторить внешний цикл. Внешний цикл организовать с помощью условного оператора. Условием окончания внешнего цикла считать $x>5$.

3.233. Составить программу решения задачи 1.152. Вывод на печать выполнять после вычисления каждого значения z.

3.7. ОРГАНИЗАЦИЯ ПОДПРОГРАММ (ПРОЦЕДУР)

При решении многих научно-технических задач встречаются однотипные участки алгоритмов. Для сокращения размера программы целесообразно выделять повторяющиеся участки в подпрограммы (отдельные процедуры). В подпрограммах описываются процессы вычисления по требуемому алгоритму при формальных параметрах. В основной программе (в соответствующих местах) производится обращение к подпрограммам при заданных значениях фактических параметров. В языке ПЛ/1 используются процедура-функция и процедура-подпрограмма (подпрограмма общего вида).

Процедура-функция

Процедура-функция используется, если в результате выполнения подпрограммы вычисляется только одна величина. Структура процедуры-функции следующая:

```
имя процедуры: PROCEDURE [(список формальных параметров)] [RETURNS
               (описатели результата)];
               оператор 1;
               оператор 2;      } операторы подпрограммы
               оператор n;      }
               RETURN (выражение);
               END [имя процедуры];
```

где имя процедуры — имя, присваиваемое программистом подпрограмме, — произвольный идентификатор, содержащий не более семи символов; первый символ — буква.

В списке формальных параметров указываются имена простых переменных, массивов, структур, точек входа в другие подпрограммы.

Если среди формальных параметров есть имя массива, то этот массив должен быть обязательно описан в процедуре в операторе DECLARE.

В тексте процедуры переменные должны быть объявлены в операторе описания DECLARE, иначе их описатели назначаются по соглашению.

Описатели, следующие после описателя RETURNS, указывают характеристики результата выполнения процедуры функции (DECIMAL или BINARY, FLOAT или FIXED, разрядность и т. п.). Если описатель RETURNS отсутствует, то характеристики результата определяются по соглашению.

Оператор RETURN (выражение) вычисляет значение указанного выражения и возвращает это значение в основную программу в качестве значения имени процедуры-функции, причем перед возвращением происходит преобразование результата в соответствии с описателями RETURNS.

В основной программе, чтобы отличить имя процедуры-функции от имени переменных, необходимо ее имя при отсутствии описателя RETURNS в описании процедуры объявить в операторе DECLARE следующего формата:

DECLARE имя процедуры ENTRY;

При наличии же описателя RETURNS в описании процедуры оператор DECLARE должен иметь формат

DECLARE имя процедуры ENTRY RETURNS (описатели);

Во втором варианте описатели RETURNS должны быть такими же, как у описателя RETURNS в описании процедуры.

Обращение к процедуре-функции записывается так:

имя процедуры (список фактических параметров).

Вызов процедуры-функции происходит автоматически при вычислении выражений, в которые входит имя процедуры-функции.

Фактическими параметрами могут быть константы, простые и индексированные переменные, массивы, выражения, имена других подпрограмм. Эти параметры должны согласовываться с формальными по типу, количеству, порядку их следования.

Для автоматического согласования описателей формальных и фактических параметров можно использовать в основной программе оператор DECLARE, в котором одновременно объявляются имя процедуры-функции и описатели формальных параметров:

DECLARE имя процедуры-функции ENTRY (список описателей соответствующих формальных параметров процедуры-функции);

или

DECLARE имя процедуры-функции ENTRY (список тот же)
RETURNS (описатель результата);

● **3.234.** Составить программу вычисления функции

$$d = r\sum_{i=1}^{100}(x_i^2+y_i^2)+q\sum_{i=1}^{50}(x_i^2+z_i^2)+t\sum_{i=1}^{100}(y_i^2+z_i^2),$$

где x_i, y_i, z_i — элементы массивов. Для вычисления суммы использовать процедуру-функцию.

Составим процедуру-функцию для вычисления суммы вида

$$s=\sum_{i=1}^{n}(a_i^2+b_i^2),$$

где a_i, b_i — элементы массивов A, B.

Формальные параметры, означающие пределы суммирования, имена суммируемых массивов чисел, обозначим *n*, A, B. Присвоим процедуре-функции имя SUM. В основной программе необходимо трижды обратиться к этой процедуре для вычисления сумм при соответствующих значениях фактических параметров:

```
      /*ОСНОВНАЯ ПРОГРАММА*/
OW: PROCEDURE OPTIONS (MAIN);
      DECLARE (X, Y, Z) (100) FIXED (4,1),
              (R, Q, T) FIXED (3);
      /*ОПИСАНИЕ ИМЕНИ ПРОЦЕДУРЫ-ФУНКЦИИ И ОПИСАТЕЛЕЙ
      ФОРМАЛЬНЫХ ПАРАМЕТРОВ И РЕЗУЛЬТАТА*/
      DECLARE SUM ENTRY (, (2) FLOAT (6)) RETURNS (FIXED (6,2));
              GET LIST (R, Q, T, X, Y, Z);
              D=R*SUM (100, X, Y))+Q*SUM (050, X, Z)+T*SUM (100,
              Y, Z);
      PUT DATA (D);
      END OW;
      /*ПОДПРОГРАММА SUM*/
SUM: PROCEDURE (N, A, B) RETURNS (DECIMAL FIXED (6,2));
      DECLARE N FIXED (3), (A(*), B(*)) FLOAT (6);
      S=0;
      DO I=1 TO N;
      S=S+A(I)*A(I)+B(I)*B(I);
      END;
      RETURN (S);
      END SUM;
```

В основной программе массивы X, Y, Z имеют описатели FIXED (4,1), а в процедуре-функции — описатели FLOAT (6). Для согласования описателей в основной программе включен оператор DECLARE SUM ENTRY с совокупностью описателей формальных параметров процедуры и результата. При обращении к процедуре-функции SUM и при передаче параметров в подпрограмму фактические параметры будут автоматически преобразованы в соответствии с описателями формальных параметров.

В основной программе для вычисления первого слагаемого (первой суммы) осуществляется обращение к процедуре по имени SUM (100, X, Y) и происходит передача фактических параметров в процедуру SUM. Выполняются операторы процедуры-функции. По оператору возврата RETURN(S) вычисленное значение результата S присваивается имени SUM и возвращается в основную программу в то место, откуда была вызвана процедура-функция. Аналогичные действия выполняются и при вычислении второй и третьей сумм.

В подпрограмме SUM используются формальные массивы A и B с описателями (A(*), B(*)) FLOAT (6), где «*» означает, что размерности массивов будут такими же, как и у передаваемых фактических параметров.

Процедура-подпрограмма

Процедура-подпрограмма используется для получения нескольких выходных результатов.

Структура процедуры-подпрограммы:

имя процедуры: PROCEDURE (список формальных параметров);

```
оператор 1;
оператор 2;
. . . . . } операторы подпрограмм
оператор n;
. . . . . .
RETURN
. . . . . .
END [имя процедуры];
```

Правила записи формальных и фактических параметров здесь такие, как и для процедуры-функции. Для согласования описателей формальных и фактических параметров в основной программе используется оператор DECLARE следующего формата:

DECLARE имя процедуры ENTRY (описатели формальных параметров процедуры-подпрограммы);

Если для какого-либо параметра нет необходимости менять описатели, то соответствующие описатели опускаются, однако все разделяющие запятые должны быть сохранены.

Обращение к процедуре-подпрограмме осуществляется по оператору CALL, имеющему формат

CALL имя процедуры (список фактических параметров);

В момент вызова процедуры-подпрограммы управление передается в подпрограмму, имя которой указано в операторе CALL, и осуществляется передача фактических параметров в подпрограмму. После выполнения вычислений в подпрограмме управление по оператору RETURN (возврат) передается в основную программу на оператор, следующий за оператором CALL.

● **3.235.** Составить программу вычисления значений корней уравнений, полагая, что корни уравнений действительные.

Уравнения имеют вид $\cos\alpha y^2+6{,}3\beta y+7{,}08=0$; $pt^2+qt+r=0$; $(f+0{,}5\cdot10^{-4})z^2+(e^v-\sin u)z=0$.

Алгоритм вычисления корней уравнения вида $ax^2+bx+c=0$ запишем в виде процедуры-подпрограммы. Формальными параметрами процедуры будут коэффициенты уравнения a, b, c и корни уравнения x_1, x_2. Здесь два выходных параметра, поэтому использовать процедуру-функцию нельзя. Присвоим заглавие процедуре-подпрограмме

KORNI : PROCEDURE (A, B, C, X1, X2);

Текст процедуры-подпрограммы запишем после основной программы.

```
    /*ОСНОВНАЯ ПРОГРАММА*/
V: PROCEDURE OPTIONS (MAIN);
   DECLARE KORNI ENTRY (5) FLOAT (6));
   DECLARE (ALPHA, BETA, P, Q, R, F, V, U) FIXED (5,2);
   GET LIST (ALPHA, BETA, P, Q, R, F, V, U);
   /*НАХОЖДЕНИЕ КОРНЕЙ УРАВНЕНИЯ Y1, Y2*/
   CALL KORNI (COS (ALPHA), 6.30*BETA, 7.08, Y1, Y2);
   /*НАХОЖДЕНИЕ КОРНЕЙ УРАВНЕНИЯ T1, T2*/
   CALL KORNI (P, Q, R, T1, T2);
   /*НАХОЖДЕНИЕ КОРНЕЙ УРАВНЕНИЯ Z1, Z2*/
   CALL KORNI (F+0,5E-4, EXP (V)-SIN (U), 0., Z1, Z2);
   PUT DATA (Y1, Y2,'__' , T1, T2,'__' , Z1, Z2);
   END;
      /*ПРОЦЕДУРА-ПОДПРОГРАММА*/
```

```
KORNI: PROCEDURE (A, B, C, X1, X2);
       DECLARE (A, B, C, X1, X2) FLOAT (6);
       D=SQRT (B**2—4*A*C);
       Z=2*A;
       X1=(—B+D)/Z;
       X2=(—B—D)/Z;
       RETURN;
       END KORNI;
```

При первом обращении к подпрограмме по оператору CALL формальные параметры A, B, C заменяются значениями фактических параметров

COS(ALPHA), 6.30*BETA, 7.08.

Выполняются операторы подпрограммы, и вычисленные значения корней присваиваются переменным Y1, Y2.

По оператору подпрограммы RETURN осуществляется возврат в основную программу к оператору, следующему за первым оператором CALL, вызвавшим подпрограмму. Поэтому далее выполняется второй оператор CALL.

При втором обращении к подпрограмме определяются значения корней T1, T2 второго уравнения, при третьем — значения корней Z1, Z2 третьего уравнения.

Поскольку значения коэффициентов уравнений в основной программе определены описателем FIXED (5, 2), а в подпрограмме — значения формальных параметров A, B, C FLOAT (6), то для согласования описателей в основной программе включен описатель DECLARE ENTRY.

Общие области памяти

С целью экономного использования памяти выделяются с помощью описателей EXTERNAL и DEFINED одни и те же ее области для хранения нескольких переменных.

Описатель EXTERNAL позволяет выделить одну и ту же область памяти двум (и более) переменным или массивам для нескольких процедур. В каждой из этих процедур общая переменная (EXTERNAL-переменная) должна быть объявлена в операторе DECLARE с описателем EXTERNAL с тем же именем и теми же описателями. Например, есть основная программа (главная процедура) A и процедуры-подпрограммы B, C:

```
A:  PROCEDURE OPTIONS (MAIN);
    DECLARE X (10, 20) FIXED (4,1) EXTERNAL;
    . . . . . .
    END A;
B:  PROCEDURE (F);
    DECLARE X (10, 20) FIXED (4,1) EXTERNAL;
    . . . . . .
    END B;
C:  PROCEDURE (M, Q);
    DECLARE X (10, 20) FIXED (4,1) EXTERNAL;
    . . . . . .
    END C;
```

Для хранения элементов массива X в основной программе и подпрограммах B и C выделяется одна общая область памяти, т. е .все три явных описания массива X задают один и тот же массив. Если бы описатель EXTERNAL отсутствовал, то описания массива X задавали бы три различных массива — один для программы и по одному в каждой подпрограмме.

Переменные с описателем EXTERNAL не могут быть формальными параметрами, что часто используется для передачи параметров в подпрограмму, которая может стать подпрограммой без параметров, т. е. передача параметров идет через общие области памяти.

Описатель DEFINED позволяет выделить одну и ту же область памяти двум (и более) переменным или массивам в пределах одной программной единицы.

Описатель имеет формат

DEFINED идентификатор базы;

Идентификатор базы — имя переменной или массива, область памяти которой используется другим идентификатором. Например,

```
DECLARE  A (100) FIXED (5),
         B (100) FIXED DEFINED A;
```

По этому описанию для массива A (100 элементов) будет выделена область памяти. Для массива B область памяти не выделяется. По описателю DEFINED оба массива A и B используют одну и ту же область памяти.

● **3.236.** Составить программу решения задачи 1.144, используя для вычисления математического ожидания и дисперсии процедуру без параметров. Для экономии памяти массивы A, B, C и формальный массив X хранить в одной области памяти.

В основной программе с помощью описателя DEFINED выделим общую память для массивов A, B, C, полагая идентификатор A базовым, а с помощью описателя EXTERNAL — общую память для массива A программы и процедуры. Чтобы процедура была без параметров, необходимо все параметры, передаваемые в процедуру и обратно, указать с описателем EXTERNAL:

```
W:  PROCEDURE OPTIONS (MAIN);
    DECLARE A(100) FIXED (6,3) EXTERNAL,
            (B (100) C (100)) FIXED (6,3) DEFINED A,
            (MX, DX) FLOAT (6) EXTERNAL,
            N BINARY FIXED EXTERNAL;
    GET LIST (N, (A(I) DO I=1 TO N));
    CALL STATIC;
    PUT DATA (MX, DX);
    GET LIST ((B(I) DO I=1 TO N)) SKIP;
    CALL STATIC;
    PUT DATA (MX, DX) SKIP (2);
    GET LIST ((C(I) DO I=1 TO N)) SKIP;
    CALL STATIC;
    PUT DATA (MX, DX) SKIP (2);
    END W;
STATIC: PROCEDURE;
    DECLARE A (100) FIXED (6,3) EXTERNAL,
            (MX, DX) FLOAT (6) EXTERNAL,
            N BINARY FIXED EXTERNAL;
    MX=0; DX=0;
B1: DO I=1 TO N;
    MX=MX+A(I);
    END B1;
    MX=MX/N;
B2: DO I=1 TO N;
    DX=DX+(A(I)-MX)**2;
    END B2;
    DX=DX/N;
    RETURN;
    END STATIC;
```

Так как массивы A, B, C хранятся в памяти в одном месте, то вводиться они должны поочередно. Параметры в подпрограмму передаются через EXTERNAL-переменные. В основной программе можно сократить запись, организовав цикл следующим образом:

```
. . . . . .
    GET LIST (N);
CIKL: DO I=1 TO 3;
    GET LIST ((A (I) DO I=1 TO N)) SKIP;
    CALL STATIC;
    PUT DATA (MX, DX) SKIP (2);
    END CIKL;
    END;
```

При выполнении первого GET будут вводиться с перфокарт элементы массива A, при выполнении второго GET — элементы массива B в ту же область памяти.

Передача имени процедуры

В ПЛ/1 существуют средства, с помощью которых одна процедура A может передавать другой процедуре B в качестве фактического параметра имя какой-то третьей процедуры C. Имя внешней процедуры C, используемое в списках фактических параметров программы A, обязательно должно быть объявлено в программе с помощью оператора

DECLARE имя внешней процедуры ENTRY;

Аналогично, в вызываемой процедуре B соответствующий формальный параметр также должен быть объявлен оператором

DECLARE имя формального параметра ENTRY;

В качестве фактических параметров могут использоваться имена математических встроенных функций — sin, cos и т. п.

Для описания этих имен в вызывающей программе необходимо вместо описателя ENTRY использовать описатель BUILTIN.

● **3.237.** Составить программу вычисления функции

$$R = \int_0^{\pi/4} \cos \alpha d\alpha + \int_0^{\pi/2} \frac{t}{\sqrt{1 - 0{,}2 \sin^2 t}}\, dt.$$

Для вычисления интеграла использовать процедуру-функцию. В процедуре-функции необходимо один раз вычислять интеграл от одной подынтегральной функции, в частности cos α, во второй раз — от другой. Опишем в подпрограмме-функции с именем SIMPS вычисление интеграла вида $\int_a^b f(x)\,dx$. Тогда при первом обращении $f(x) = \cos \alpha$, при втором $f(x) = t\sqrt{1 - 0{,}2 \sin^2 t}$.

Для вычисления значения функции $f(x)$ воспользуемся при первом обращении стандартной математической функцией с именем COS, при втором — подпрограммой-функцией FI с формальным параметром t, вычисляющей значение $t/\sqrt{1 - 0{,}2 \sin^2 t}$.

Имена функций COS, FI должны передаваться в подпрограмму-функцию в качестве фактических параметров. Поэтому они должны быть указаны описателем COS BUILTIN, FI ENTRY.

Для вычисления интеграла $\int_a^b f(x)dx$ воспользуемся методом прямоугольников, разбивая интервал интегрирования на n частей с шагом h. Описатели переменным присвоим по соглашению.

Значения шагов интегрирования h_1, h_2 для вычисления соответственно первого и второго интегралов будем вводить с перфокарт в основной программе.

Программа имеет вид

```
    /*ОСНОВНАЯ ПРОГРАММА*/
W: PROCEDURE OPTIONS (MAIN);
    DECLARE FI ENTRY,
             COS BUILTIN,
             SIMPS ENTRY ((4) FLOAT (6));
    GET LIST (H1, H2);
    R1=SIMPS (0, 3.1415/4, H1, COS);
    R2=SIMPS (0, 3.1415/2, H2, FI);
    R=R1+R2;
    PUT DATA (R);
    END W;
    /*ПОДПРОГРАММА-ФУНКЦИЯ ВЫЧИСЛЕНИЯ ИНТЕГРАЛА*/
SIMPS: PROCEDURE (A, B, H, F);
       DECLARE F ENTRY;
       S=0;
  CIKL: DO X=A BY H WHILE (X<B);
          S=S+FI(X);
          END CIKL;
       S=S*H; RETURN (S);
       END SIMPS;
       /*ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ФУНКЦИИ FI*/
    FI: PROCEDURE (T);
       RETURN (T/SQRT (1-0.2*SIN (T)**2));
       END FI;
```

Описание точки входа в подпрограмму SIMPS дано совокупностью описателей FLOAT (6) для того, чтобы при передаче констант 0, $\pi/4$, $\pi/2$ они были представлены в формате с плавающей точкой.

При первом обращении к процедуре SIMPS вместо формальных параметров A, B будут переданы значения 0 и $\pi/4$, вместо шага H — значение H1 и вместо функции F — имя стандартной функции COS. После выполнения операторов подпрограммы полученный результат присваивается имени процедуры SIMPS и передается в основную программу. При втором обращении будет вычислено значение второго интеграла. При вычислении значения функции FI используются также встроенные математические функции SQRT и SIN, но они не описываются в операторе DECLARE, так как не являются фактическими параметрами.

▲ **3.238.** Составить программу решения задачи 2.196, используя процедуру-функцию.

▲ **3.239.** Составить программу решения задачи 2.200, используя процедуру-функцию.

▲ **3.240.** Составить программу решения задачи 2.201, используя процедуру-функцию.

▲ **3.241.** Составить программу решения задачи 2.202, используя процедуру-функцию.
▲ **3.242.** Составить программу решения задачи 2.204, используя процедуру-функцию.
▲ **3.243.** Составить программу решения задачи 2.205, используя процедуру-подпрограмму.
▲ **3.244.** Составить программу решения задачи 2.206, используя процедуру-подпрограмму.
▲ **3.245.** Составить программу решения задачи 2.207, используя процедуру-подпрограмму.
▲ **3.246.** Составить программу решения задачи 2.208, используя процедуру-подпрограмму.
▲ **3.247.** Составить программу решения задачи 2.209; используя процедуру-функцию и процедуру-подпрограмму.
▲ **3.248.** Составить программу решения задачи 2.210, используя процедуру-подпрограмму без передачи параметров.

ЛИТЕРАТУРА

1. *Бухтияров А. М., Фролов Г. Д., Олюнин В. Ю.* Сборник задач по программированию на языке ПЛ/1. — М.: Наука, 1978, 320 с.
2. Вычислительная техника в инженерных и экономических расчетах/*Петров А. В., Алексеев В. Е.* и др.; Под ред. *Б. В. Анисимова.* — М.: Высшая школа, 1975, 302 с.
3. *Дайитбегов Д. М., Черноусов Е. А.* Основы алгоритмизации и алгоритмические языки. — М.: Статистика, 1979, 375 с.
4. *Лепин-Дмитрюков Г. А., Овчаренко Е. К.* Сборник задач по программированию на языке ПЛ/1. — М.: Советское радио, 1980, 304 с.
5. Программирование на ПЛ/1 ОС ЕС/*Аугустон М. И., Балодис Р. П., Барздинь Я. М.* и др. — М.: Статистика, 1979, 269 с.
6. Сборник задач по программированию на Фортране/*Алексеев В. Е., Борисов С. В.* и др. — М.: МВТУ, 1978, 91 с.
7. Фортран ЕС ЭВМ/*Бриг З. С., Капилевич Д. В., Котик С. Ю., Цагельский В. И.* — М.: Статистика, 1978, 264 с.
8. *Фурунжиев Р. И.* Вычислительные машины и программирование. — Минск: Вышейшая школа, 1981, 239 с.

2.1

1), 2), 3),

2.2

1) 375 2) -64 3) 645000 4) 400 5) -0 6) 100000000

2.3

а) 1) 0.0001 2) -297.3753 3) 0.333333 4) 9.120340 5) 0,000000
6) 0.008971 7) 1980.0 8) -0.6 9) 8.765876 10) 0.000003
11),12) НЕЛЬЗЯ ПРЕДСТАВИТЬ

б) 1) 1.E-4 2) -0.297375E3 3) 0.333333E1 4) 0.9120340E1 5) 0.2953E-11
6) 0.89710E-2 7) 0.198E4 8) -.6E0 9) 0.876588E1 10) 0.257E-5
11) 0.197813E14 12) 0.666667E12

в) 1) 1.D-4 2) -0.297375256D3 3) 0.33333333333333D1 4) 0.91203401D1
5) 0.2953D-11 6) 0.897107510-2 7) 0.198D4 8) -0.600 9(0.87658765D1
10) 0.257D-5 11) 0.197813D14 12) 0.666666666666667D12

2.4

1), 4), 7), 8), 10), 11)

2.5

1) INTEGER R,Q,S*2(10)
   REAL*8 X,Y(50)
2) INTEGER*2 I,K,N*4(10)
   LOGICAL C

3) COMPLEX*16 D(40)
   REAL K

2.6

1) INTEGER*2 I,J,K,L,KOL*4(15)
   REAL*8 A,B,C,D,E,F
2) IMPLICIT INTEGER*2 (A-K)
   REAL*8 D1(20)
3) DIMENSION X(30) ,Y(30)
   REAL*8 XSO

2.7

1) I,Q-ЦЕЛЫЕ ПЕРЕМЕННЫЕ НЕСТАНДАРТНОЙ ДЛИНЫ,I1,L,N1;-ЦЕЛЫЕ ПЕРЕМЕННЫЕ СТАНДАРТНОЙ ДЛИНЫ,МАССИВЫ NOM,KOL,R-ЦЕЛЫЕ СТАНДАРТНОЙ ДЛИНЫ,МАССИВ Q1-ЦЕЛЫЙ НЕСТАНДАРТНОЙ ДЛИНЫ,ПЕРЕМЕННЫЕ LST,X1 И МАССИВ LS-ДЕЙСТВИТЕЛЬНЫЕ СТАНДАРТНОЙ ДЛИНЫ,ПЕРЕМЕННАЯ NS И МАССИВ X -ЦЕЛЫЕ НЕСТАНДАРТНОЙ ДЛИНЫ,3) ПЕРЕМЕННАЯ INT И МАССИВ E-ДЕЙСТВИТЕЛЬНЫЕ СТАНДАРТНОЙ ДЛИНЫ,ПЕРЕМЕННАЯ I И МАССИВ IN-ЦЕЛЫЕ СТАНДАРТНОЙ ДЛИНЫ,D1-ДЕЙСТВИТЕЛЬНАЯ ПЕРЕМЕННАЯ СТАНДАРТНОЙ ДЛИНЫ,D-ЛОГИЧЕСКИЙ МАССИВ СТАНДАРТНОЙ ДЛИНЫ.

2.8

1) X/(1.+X*X/(2*Y))
2) (A+B*X)/(C*D) ИЛИ (A+B*X)/C/D
3) EXP(X)-SIN(X)**2
4) 1.+ATAN(X/(1.+SQRT(X)))
5) X**Y**Z
6) (A*SQRT(X)-B*B*X**(1./3.))/(X+0.5)**(1./5.)
7) A*X**3-B*X*X+A*COS(ABS(X))**3
8) (SIN(X)**3-COS(X**3))/(2*X)-1.E-3
9) (X**M-A**M)/SQRT(A*X)
10) EXP(-A*X)*SIN(OMEGA*T+FI)
11) ALOG(ABS(X-A))/(X*X-A*X+A*A)
12) X**(1./3.)-2.*X**(2./5.)+EPS

2.9

1),2),3) ДЕЙСТВИТЕЛЬНОЕ СТАНДАРТНОЙ ДЛИНЫ 4) ЦЕЛОЕ СТАНДАРТНОЙ ДЛИНЫ 5) КОМПЛЕКСНОЕ НЕСТАНДАРТНОЙ ДЛИНЫ 6) ДЕЙСТВИТЕЛЬНОЕ НЕСТАНДАРТНОЙ ДЛИНЫ.

2.10

1) .FALSE. 2) .TRUE.3) .FALSE. 4) .TRUE. 5) .TRUE. 6) .TRUE.

2.11

1) а) X.NE.0.AND.Y.NE.0
   б) X.NE.0.AND.Y.GT.0
   в) X.GE.0.AND.Y.GT.0
   г) B*B-4*A*C.GE.0
   д) ABS(X).LE.1.AND.ABS(Y).LE.1
   е) Y.GT.0.AND.X.NE.Y
2) N/3*3-N.EQ.0
3) A/5*5-A.EQ.0.AND.B/5*5-B.EQ.0

2.12

```
1) READ(5,1) NAME,KOL,A
 1 FORMAT(I1,I3,F7.2)
   WRITE(6,2) NAME,KOL,A
 2 FORMAT(1H ,I2.I4,F8.2)
3) READ(5,1) A,B,C
 1 FORMAT(3F4.1)
   WRITE(6,2) A,B,C
 2 FORMAT(3H A=,F4.1,3H B=,
  *F4.1,3H C=,F4.1)
5) READ(5,1) A,B,C
 1 FORMAT(F3.0,F3.1,E4.0)
   WRITE(6,2) A,B,C
 2 FORMAT(' ИСХОДНЫЕ ДАННЫЕ'
  :2F4.1,E7.1)
```

```
2) READ(5,1) X,Y,Z
 1 FORMAT(2F7.3,E7.2)
   WRITE(6,2) X,Y,Z
 2 FORMAT(1H ,2F8.3,E8.2)
4) READ(5,1) E,F,D
 1 FORMAT(E9.3,D11.3,D8.3)
   WRITE(6,2) E,F,D
 2 FORMAT(1H ,E10.3,D13.3,D10.3)
```

2.13

```
1) READ(5,1) X,Y
 1 FORMAT(F8.6,F8.0)
3) READ(5,1) K,N,X
 1 FORMAT(I3,I2,E6.2)
5) WRITE(6,2) X1,X2.
 2 FORMAT(16H КОРНИ УРАВНЕНИЯ
  */4H X1=,F7.5,4H X2=,F7.5)
```

```
2) READ(5,1) Z,W
 1 FORMAT(E10.5,D9.5)
4) WRITE(6,2)
 2 FORMAT(  РЕЗУЛЬТАТЫ СЧЕТА'/
6) WRITE(6,2)
 2 FORMAT(  АРГУМЕНТ X
  *     ФУНКЦИЯ Y')
```

2.14

```
   DIMENSION C(200)
   READ(5,1) C
 1 FORMAT(16F5.1)
```

2.15

```
   DIMENSION X(100),Y(100)
   READ(5,   X,Y
 1FORMAT(10F8.3)
```

2.16

```
   DIMENSION D(80)
   READ(5,1) A,B,C,D
 1 FORMAT(F5.2,F6.2,F7.2/(13F6.2))
```

2.17

```
   DIMENSION D(120)
   READ(5,1) M,(D(I),I=1,M)
 1 FORMAT(I3/(13F6.2))
```

2.18

```
   DIMENSION A(30,30)
   READ(5,1) N,M,((A(I,J),I=1,N),
  *J=1,M)
 1 FORMAT(2I2/(16F5.1))
```

2.19

```
   DIMENSION X(100,100)
   READ(5,1) A,B,N,((X(I,J),
  *J=1,N),I=1,N)
 1 FORMAT(2F6.3,I2/(20F4.2))
```

2.20

```
   DIMENSION M(120)
   READ(5,1) K,(M(I),I=1,K)
 1 FORMAT(26I3)
```

2.21

```
   DIMENSION NOM(99),X(99)
   READ(5,1) N,(NOM(I),X(I),I=1,N)
 1 FORMAT(I2/(I5/F5.2))
```

2.22

```
   DIMENSION NOM(99),X(99)
   READ(5,1) N,(NOM(I),I=1,N)
 1 FORMAT(I2/(16I5))
   READ(5,2) (X(I),I=1,N)
 2 FORAMT(16F5.2)
```

2.23

```
   DIMENSION Z(50)
   .
   .
   WRITE(6,3) Z
 3 FORMAT(10F12.4)
```

2.24

```
   DIMENSION X(20),Y(20)
   .
   .
   WRITE(6,3) X,Y
 3 FORMAT(10E12.4)
```

2.25

```
   DIMENSION X(10),Y(10)
   .
   .
   DO 2 I=1,10
 2 WRITE(6,3) X(I),Y(I)
 3 FORMAT(1H ,F6.2,E12.3)
```

2.26

```
   DIMENSION Y(100)
   .
   .
   WRITE(6,4) M,(Y(I),I=1,M)
 4 FORMAT(I5/(8E15.5))
```

2.27

```
   DIMENSION A(20),B(20),C(20)
   .
   .
   DO 2 I=1,20
 2 WRITE(6,3) I,A(I),B(I),C(I)
 3 FORMAT(I3,3E20.5)
```

2.28

```
   DIMENSION A(10,8)
   .
   .
   WRITE(6,3) ((A(I,J),J=1,8),
  *I=1,10)
 3 FORMAT(1H ,8E13.4)
```

2.29

```
   DIMENSION A(30,30)
   .
   .
   WRITE(6,4) ((A(I,J),J=1,30),
  *I=1,30)
 4 FORMAT(10E12.3)
```

2.30

```
   DIMENSION Z(15,8)
   .
   .
   DO 2 I=1,15
 2 WRITE(6,3) I,(Z(I,J),J=1,8)
 3 FORMAT(I3,8E12.3)
```

2.31

```
   DIMENSION Z(15,8),NOM(8)
   .
   .
   WRITE(6,3) (NOM(J),J=1,8)
 3 FORMAT(9X,I1,7(11X,I1))
   DO 2 I=1,15
 2 WRITE(6,4) I,(Z(I,J),J=1,8)
 4 FORMAT(I3,8E12.3)
```

2.32

```
   DIMENSION X(10,10)
   .
   .
   DO 2 I=1,10
 2 WRITE(6,3) (X(I,J),J=1,I)
 3 FORMAT(1H ,10F8.2)
```

2.33

```
   DIMENSION X(10,10)
   .
   .
   DO 2 I=1,10
 2 WRITE(6,3) (X(I,J),J=I,10)
 3 FORMAT(1H ,10F8.2)
```

2.34

```
   DIMENSION A(10,10)
   .
   .
   WRITE(6,2) (A(I,I),I=1,10)
 2 FORMAT(10E12.3)
```

2.35

```
      DIMENSION C(8,5)

      WRITE(6,2) ((C(I,J),J=1,5),
     *I=1,8)
    2 FORMAT('ЗНАЧЕНИЯ МАТРИЦЫ C'/
     *(5E20.5))
```

2.36

```
    1) C=(A*X+B**3)/SQRT((A-B)**3)
    2) R=X/A-1./3.14159*ALOG(A+B*EXP(P*X))
       3) T=1/X**2*(Y/1E2)**X
       4) Z=2*SIN(3.14159*N-X)**3
       5) P=((((A(6)*X+A(5))*X+A(4))*X+A(3))*X+A(2))*X+A(1)
       6) E=A.GT.B
       7) F=A*A+1.GE.0.5
       8) D=A.GT.0.AND.B.GT.0
       9) W=D*D-4*A*C.GE.0.OR.A.NE.0
      10) U=(A.EQ.B.OR.C.EQ.D.AND.(A+B).GT.2*C).AND..NOT.A.GT.C
```

2.39

```
      DIMENSION X(3),Y(3)
      REAL M(3),M1
      READ(5,1) X,Y,M
    1 FORMAT(9F4.1)
      M1=M(1)+M(2)+M(3)
      XC=(M(1)*X(1)+M(2)*X(2)+M(3)*X(3))/M1
      YC=(M(1)*Y(1)+M(2)*Y(2)+M(3)*Y(3))/M1
      WRITE(6,2) XC,YC
    2 FORMAT(1H ,F4.1,1X,F4.1)
```

3.71

```
DCL A(10,10)FIXED(7,2); DO I=1 TO 10; PUT EDIT((A(I,J) DO J=1 TOI))
(SKIP,10(F(8,2),X(4))); END;
```

3.73

```
A(1)=1 A(2)=3 A(3)=5 A(4)=7 A(5)=9 A(6)=0 A(7)=0 A(8)=0 A(9)=0 A(10)=0
```

3.74

```
GET LIST(((A(I,J) DO I=1 TO 5)DO J=2 TO3));
```

3.75

```
1) PUT DATA((X(J),Y(J),Z(J) DO J=1 TO 10 BY 1));
2) PUT DATA ('(J)',Y(J),Z(J)DO J=10 TO 1 BY-1));
4) PUT DATA  X(J),Y(J)DO J=1 TO 10),(Z(J)DO J=1 TO 10));
```

3.76

```
A(0,0)=1E-13  A(0,1)=1E-13  A(0,2)=1E-13
A(1,2)=18E-15 A(3,4)=18E-15   A(2,3)=18E-15
```

3.78

```
1)   Y=SIN(A)**2+COS(A-3.1415)+1;
2)   R=(X(I)**2+X(2*I)-C)/(X(I)-B)+B;
3)   Y=0.693147+2*(A/(4 +A**3 /(4 +A)**3+A**5/5 /(4 +A)**5));
4)   W=A A+1>=0.7;      5)   0=A>0&B>0
```

2.44

```
      READ(5,1)A,S
    1 FORMAT(2F5.1)
      X=-A/2.+SQRT(A*A/4.+2.*S)
      WRITE(6,2)X
    2 FORMAT(F7.2)
      STOP
      END
```

2.47

```
      READ(5,1)A,B,C
    1 FORMAT(3F4.1)
      D=B*B-4.*A*C
      Z=2.*A
      E=-B/Z
      F=SQRT(ABS(D))/Z
      IF(D)2,3,3
    2 WRITE(6,4)E,F
      GOTO 5
    3 X1=E+F
      X2=E-F
      WRITE(6,4)X1,X2
    4 FORMAT(2E15.5)
    5 STOP
      END
```

2.49 a)

```
      READ(5,1)X
    1 FORMAT(F4.2)
      IF(X+1.)2,2,3
    2 Y=0.
      GOTO 4
    3 Y=1.-ABS(X)
    4 WRITE(6,5)Y
    5 FORMAT(E12.4)
      STOP
      END
```

2.49 b)

```
      READ(5,1)R,X
    1 FORMAT(2F4.2)
      IF(R+X)2,2,3
    2 Y=0.
      GOTO 4
    3 IF(X-2.*R*SQRT(2.)/3.)5,5,6
    5 Y=SQRT(R*R-X*X)
      GOTO 4
    6 Y=R/3.
    4 WRITE(6,7)Y
    7 FORMAT(E12.4)
      STOP
      END
```

2.50

```
      LOGICAL C
      READ(5,1)X
    1 FORMAT(F5.2)
      C=.FALSE.
      IF(X.GT.0.)C=.TRUE.
      WRITE(6,2)C
    2 FORMAT(3H C=,L4)
      STOP
      END
```

2.51

```
      READ(5,1)A,B
    1 FORMAT(2F5.1)
      IF(A-B)2,2,3
    2 N=2
      Z=B*B
      GOTO 4
    3 N=1
      Z=A*A
    4 WRITE(6,5)Z,N
    5 FORMAT(F10.2,I2)
      STOP
      END
```

2.52

```
      READ(5,1)R,X0,Y0
    1 FORMAT(3F5.2)
      N=0
      IF(SQRT(X0*X0+Y0*Y0).LE.R)N=1
      WRITE(6,2)N
    2 FORMAT(I2)
      STOP
      END
```

2.54 a)

```
      READ(5,1)X,Y
    1 FORMAT(2F5.2)
      IF(X)2,3,3
    3 IF(Y)4,5,5
    4 N=4
      GOTO 6
    5 N=1
      GOTO 6
    2 IF(Y)7,8,8
    7 N=3
      GOTO 6
    8 N=2
    6 WRITE(6,9)N
    9 FORMAT(I2)
      STOP
      END
```

2.54 b)

```
      READ(5,1)X,Y
    1 FORMAT(2F5.2)
      N=1
      IF(X.GE.0.AND.Y.LT.0)N=4
      IF(X.LT.0.AND.Y.LT.0)N=3
      IF(X.LT.0.AND.Y.GE.0)N=2
      WRITE(6,2)N
    2 FORMAT(I2)
      STOP
      END
```

2.55

```
      READ(5,1)X
    1 FORMAT(F3.1)
      NX=5
      IF(X.LT.4.5)NX=X+0.5
      WRITE(6,2)NX
    2 FORMAT(I2)
      STOP
      END
```

2.56

```
      INTEGER X
      READ(5,1)X
    1 FORMAT(I3)
      IF(X/3*3-X)2,3,2
    2 WRITE(6,4)
    4 FORMAT(4H HET)
      GOTO 5
    3 WRITE(6,6)X
    6 FORMAT(I4)
    5 STOP
      END
```

2.61

```
      DIMENSION A(25)
      READ(5,1)A
    1 FORMAT(16F5.2)
      DO2I=1,25
    2 IF(A(I).LT.0.)A(I)=0.
      WRITE(6,3)A
    3 FORMAT(13F8.2)
      STOP
      END
```

2.63

```
      READ(5,1)X,A
    1 FORMAT(2F4.2)
      K=1
    3 Z=X**K/K**2
      WRITE(6,2)Z
      K=K+1
      IF(Z.GT.A)GOTO 3
    2 FORMAT(E12.4)
      STOP
      END
```

2.64

```
      READ(5,1)X,A
    1 FORMAT(2F4.2)
      K=1
    4 IF(X**K/K**2-A)3,2,2
    2 K=K+1
      GO TO 4
    3 WRITE(6,5)K
    5 FORMAT(I5)
      STOP
      END
```

2.66

```
      DIMENSION A(50)
      READ(5,1)A
    1 FORMAT(10F8.3)
      DO 2 I=1,50
      IF(A(I))3,2,2
    2 CONTINUE
    3 WRITE(6,4)A(I),I
    4 FORMAT(F10.3,I3)
      STOP
      END
```

2.67

```
      DIMENSION Y(100)
      READ(5,1)Y
    1 FORMAT(16F5.1)
      DO 2 I=1,100
    2 IF(Y(I).GT.0.AND.Y(I).LT.1)
     *WRITE(6,3)I
    3 FORMAT(I4)
      STOP
      END
```

2.69

```
      X=2.
    1 Y=X-ATAN(X)-3.14159
      X=X+0.1
      IF(Y.LT.0)GOTO 1
      WRITE(6,2)X
    2 FORMAT(F4.1)
      STOP
      END
```

2.70

```
      DIMENSION X(100),Y(100)
      READ(5,1)X,Y,R
    1 FORMAT(20F4.1)
      DO 2 I=1,100
    2 IF(X(I)**2+Y(I)**2.LE.R*R)
     *WRITE(6,3)I
    3 FORMAT(I4)
      STOP
      END
```

2.71

```
      READ(5,1)A,B0,BM
    1 FORMAT(3F4.1)
      X=B0
      Y=X-EXP(-(A*X)**2/2.)
      N=1
      IF(Y.LT.0.) N=-1
    2 X=X+0.2
      Y=X-EXP(-(A*X)**2/2.)
      IF(N*Y.GT.0) GO TO 2
      WRITE(6,3)X
    3 FORMAT(F5.1)
      STOP
      END
```

2.73

```
      READ(5,1)R,A4
    1 FORMAT(2F6.2)
      N=0
      R1=R*R
    5 N=N+1
      A2N=SQRT(2.*R1-2.*R*SQRT(R1-A4**2/4.))
      A4=A2N
      IF(N.LT.5) GO TO 5
      WRITE(6,2)A2N
    2 FORMAT(E14.5)
      STOP
      END
```

2.74

```
      DIMENSION X(30,30)
      READ(5,1)N,((X(I,J),I=1,N),J=1,N)
    1 FORMAT(I2/(10F8.3))
      DO 2 I=1,N
    2 IF(X(I,J).GT.0)WRITE(6,3)X(I,I)
    3 FORMAT(F9.3)
      STOP
      END
```

2.76

```
      READ(5,1)N
    1 FORMAT(I2)
      A=0.112
      C=0.112
      B=SQRT(1.-A*A)
      N1=N-1
      DO 2 I=2,N1
      D=SQRT(1.-C*C)
    2 C=C*A-D*B
      WRITE(6,3)C
    3 FORMAT(E12.4)
      STOP
      END
```

2.77

```
      READ(5,1)N
    1 FORMAT(I9)
      K=0
    2 N=N/10
      K=K+1
      IF(N.NE.0) GO TO 2
      WRITE(6,3)K
    3 FORMAT(I3)
      STOP
      END
```

2.78

```
      INTEGER X(50)
      READ(5,1)N,(X(I),I=1,N)
    1 FORMAT(I2/(20I4))
      DO 2 I=1,N
    2 IF(X(I)/3*3.EQ.X(I))WRITE(6,3)X(I)
    3 FORMAT(I5)
      STOP
      END
```

2.81

```
      DIMENSION C(11)
      READ(5,1)C
    1 FORMAT(11F6.2)
      A=0.
      B=1.
      DO 2 I=1,11
      Z=(A+B+C(I))/I
      WRITE(6,3)Z
      A=A+0.1
    2 B=B+0.2
    3 FORMAT(E11.3)
      STOP
      END
```

2.84

```
      READ(5,1)X,H
    1 FORMAT(2F5.1)
      DO 2 N=3,41,2
      X=X+H
      Y=X/N
    2 WRITE(6,3)Y
    3 FORMAT(E12.4)
      STOP
      END
```

2.85

```
      INTEGER DN
      READ(5,1)X0,DX,DN,M,N0
    1 FORMAT(2F5.1,3I2)
      X=X0+DX
      K0=N0+3*DN
      K1=2*DN
      DO 2 N=K0,M,K1
      Y=X/N
      X=X+2*DX
    2 WRITE(6,3)Y
    3 FORMAT(E12.4)
      STOP
      END
```

2.87

```
      DIMENSION Z(21)
      READ(5,1)A,B,C
    1 FORMAT(3F4.1)
      X=0.
      DO 2 I=1,21
      Z(I)=A*EXP(B*X-C*X*X)
    2 X=X+0.1
      WRITE(6,3)Z
    3 FORMAT(1H ,11E10.2)
      STOP
      END
```

2.89

```
      DIMENSION X(20),Y(20)
      READ(5,1)X
    1 FORMAT(10F8.3)
      DO 2 I=1,20
      IF(X(I))3,4,5
    3 Y(I)=-1.
      GO TO 2
    4 Y(I)=0.
      GO TO 2
    5 Y(I)=X(I)
```

```
2  CONTINUE
   WRITE(6,6)Y
6  FORMAT(10F10.3)
   STOP
   END
```

2.91

```
   DIMENSION X(100),Y(100)
   READ(5,1)X
1  FORMAT(10F8.2)
   K=0
   DO 2 I=1,100
   IF(X(I))2,2,3
3  K=K+1
   Y(K)=X(I)
2  CONTINUE
   WRITE(6,4)(Y(I),I=1,K)
4  FORMAT(10F10.2)
   STOP
   END
```

2.93

```
   DIMENSION X(100),Y(10)
   READ(5,1)X
1  FORMAT(20F4.1)
   K=0
   DO 2 I=1,100
   IF(X(I))2,2,3
3  K=K+1
   Y(K)=X(I)
   IF(K-10)2,4,4
2  CONTINUE
4  WRITE(6,5)(Y(I),I=1,K)
5  FORMAT(10F5.1)
   STOP
   END
```

2.94

```
   DIMENSION Z(100)
   READ(5,1)N,X0,XM,H
1  FORMAT(I2,3F3.1)
   X=X0
   K=0
   M=(XM-X0)/H+1.1
   DO 2 I=1,M
   Y=N*SIN(X)-COS(N*X)
   IF(Y.LT.0.OR.Y.GT.1) GO TO 2
   K=K+1
   Z(K)=Y
2  CONTINUE
   WRITE(6,3)(Z(I),I=1,K)
3  FORMAT(10E12.4)
   STOP
   END
```

2.96

```
   DIMENSION A(100),B(50),C(50)
   READ(5,1)A
1  FORMAT(10F8.3)
   DO 2 I=1,50
   B(I)=A(2*I)
2  C(I)=A(2*I-1)
   WRITE(6,3)B,C
3  FORMAT(10F10.3)
   STOP
   END
```

2.97

```
   DIMENSION Z(100)
   READ(5,1)A,B,C
1  FORMAT(3F4.1)
   X=0.
   DO 2 I=1,101
   Y=-X**3*A*X*X+B*X+C
   IF(Y.LT.0) GO TO 3
   Z(I)=Y
2  X=X+0.1
3  K=I-1
   WRITE(6,4)(Z(I),I=1,K)
4  FORMAT(10E12.3)
   STOP
   END
```

2.98

```
   DIMENSION X(60),Y(60),Z(60)
   READ(5,1)X
1  FORMAT(15F5.2)
   K=0
   N=0
   DO 2 I=1,60
   IF(X(I))3,2,4
3  N=N+1
   Z(N)=X(I)
   GO TO 2
4  K=K+1
   Y(K)=X(I)
2  CONTINUE
   WRITE(6,5)(Y(I),I=1,K)
  *(Z(I),I=1,N)
5  FORMAT(20F6.2)
   STOP
   END
```

2.108

```
   READ(5,1)X
1  FORMAT(F6.2)
   S=1.
   Y=1.
   DO 2 I=1,19
   Y=Y*X
2  S=S+Y/I
   WRITE(6,3)S
3  FORMAT(E12.4)
   STOP
   END
```

2.109

```
   DIMENSION C(30)
   READ(5,1)C
1  FORMAT(10F8.2)
   S=0.
   N=0
   DO 2 I=1,30
   IF(C(I))3,2,2
3  S=S+C(I)
   N=N+1
2  CONTINUE
   S=S/N
   WRITE(6,4)S
4  FORMAT(E12.4)
   STOP
   END
```

2.110

```
   DIMENSION A(80)
   READ(5,1)A
1  FORMAT(15F5.2)
   S=0
   N=0
   DO 2 I=1,80
   IF(A(I).LT.1.OR.A(I).GT.2)GO TO 2
   S=S+A(I)
   N=N+1
2  CONTINUE
   IF(N)5,5,4
5  S=0.
   GO TO 6
4  S=S/N
6  WRITE(6,3)S
3  FORMAT(E12.4)
   STOP
   END
```

2.112

```
   READ(5,1)N
1  FORMAT(I2)
   NFACT=0
   IF(N)5,3,4
3  NFACT=1
   GO TO 5
4  NFACT=1
   DO 2 I=1,N
2  NFACT=NFACT*I
5  WRITE(6,6)NFACT
6  FORMAT(I12)
   STOP
   END
```

2.119

```
   DIMENSION Y(25)
   READ(5,1)Y,A
1  FORMAT(13F6.2)
   Z=1.
   N=0
   DO 2 I=1,25
   IF(Y(I).LE.A) GO TO 2
   Z=Z*Y(I)
   N=N+1
2  CONTINUE
   Z=Z**(1./N)
   WRITE(6,3)Z
3  FORMAT(E12.4)
   STOP
   END
```

2.120

```
      DIMENSION A(40)
      READ(5,1)A
    1 FORMAT(10F8.2)
      S=1
      N=0
      DO 2 I=2,40,2
      IF(A(I))2,2,3
    3 S=S*A(I)
      N=N+1
      IF(N)5,5,6
    5   S=0.
      GO TO 7
    6 S=S**(1./N)
    2 CONTINUE
    7 WRITE(6,4)S
    4 FORMAT(E12.4)
      STOP
      END
```

2.121

```
      READ(5,1)X,N
    1 FORMAT(F6.3,I3)
      NF=1
      DO 2 I=1,N
    2 NF=NF*I
      S=(X/2)**N/NF
      Y=S
      DO 3 K=1,100
      Y=Y*(-X*X)/(K*(K+N))/4.
    3 S=S+Y
      WRITE(6,4)S
    4 FORMAT(E15.7)
      STOP
      END
```

2.123

```
      INTEGER X(60)
      READ(5,1)X
    1 FORMAT(20I4)
      N=0
      DO 2 I=1,60
    2 IF(X(I)/3*3.EQ.X(I))N=N+1
      WRITE(6,3)N
    3 FORMAT(I4)
      STOP
      END
```

2.124

```
      INTEGER A(75),S
      READ(5,1)A
    1 FORMAT(20I4)
      S=0
      DO 2 I=1,75
    2 S=S+A(I)
      IF(S/2*2-S)3,4,3
    3 WRITE(6,5)
    5 FORMAT(' НЕТ')
      GO TO 6
    4 WRITE(6,7)
    7 FORMAT(' ДА')
    6 STOP
      END
```

2.125

```
      DIMENSION X(100)
      READ(5,1)X
    1 FORMAT(16F5.2)
      M=0
      DO 2 I=1,100
      N=X(I)+0.5
    2 IF(N.EQ.1)M=M+1
      WRITE(6,3)M
    3 FORMAT(I3)
      STOP
      END
```

2.134

```
      READ(5,1)X,N
    1 FORMAT(F5.1,I2)
      Y=X**N
      Z=Y
      DO 2 I=2,1000
      Z=Z*(I-1)/(I*X)
      IF(Z.LT.1.E-6) GO TO 3
    2 Y=Y+Z
    3 WRITE(6,4)Y
    4 FORMAT(E15.7)
      STOP
      END
```

2.138

```
      READ(5,1)X
    1 FORMAT(F4.1)
      Z=1.
      DO 2 N=2,9
    2 Z=Z*X+N
      WRITE(6,3)Z
    3 FORMAT(E10.2)
      STOP
      END
```

2.139

```
      READ(5,1)X
    1 FORMAT(F5.2)
      S=1.
      DO 2 I=1,8
    2 S=S*X*I/(9-I)+1.
      WRITE(6,3)S
    3 FORMAT(E12.4)
      STOP
      END
```

2.140

```
      READ(5,1)X
    1 FORMAT(F4.2)
      Z=1.
      DO 2 N=1,12
    2 Z=Z*X/(13-N)+1.
      WRITE(6,3)Z
    3 FORMAT(E15.5)
      STOP
      END
```

2.147

```
      READ(5,1)A,B,C,D
    1 FORMAT(4F4.1)
      YMAX=-1.E30
      YMIN=1.E30
      X=-10.
      DO 2 I=1,101
      Y=A*X**3+B*X*X+C*X+D
      IF(YMAX.GE.Y) GO TO 3
      YMAX=Y
      GO TO 2
    3 IF(YMIN.LE.Y) GO TO 6
      YMIN=Y
    2 X=X+0.2
    6 WRITE(6,7)YMAX,YMIN
    7 FORMAT(2E15.5)
      STOP
      END
```

2.145

```
      READ(5,1)A,B,C,X0,XM,H
    1 FORMAT(6F4.1)
      Y=A*EXP(A*X0)-B*EXP(-C*X0)
      X=X0+H
      YM=A*EXP(A*X)-B*EXP(-C*X)
      N=1
      IF(Y.LT.YM) N=-1
      X=X0+2*H
      M=(XM-X)/H+1
      DO 2 I=1,M
      Y=A*EXP(A*X)-B*EXP(-C*X)
      IF(N*Y.GE.N*YM) GO TO 3
      YM=Y
    2 X=X+H
    3 WRITE(6,4)YM,N
    4 FORMAT(E12.4,I3)
      STOP
      END
```

2.150

```
      DIMENSION X(50)
      READ(5,1)X
    1 FORMAT(10F8.3)
      XMAX=X(1)
      NMAX=1
      XMIN=X(1)
      NMIN=1
      DO 2 I=2,50
      IF(X(I).LE.XMAX) GO TO 3
      XMAX=X(I)
      NMAX=I
      GO TO 2
    3 IF(X(I).GE.XMIN) GO TO 2
      XMIN=X(I)
      NMIN=I
    2 CONTINUE
      X(NMAX)=1
      X(NMIN)=-1
      WRITE(6,4)X
    4 FORMAT(10F10.3)
      STOP
      END
```

2.152

```
      DIMENSION A(80)
      READ(5,1)A
    1 FORMAT(20F4.2)
      AMAX=ABS(A(1)-A(2))
      AMIN=AMAX
      DO 2 I=2,79
      AM=ABS(A(I)-A(I+1))
      IF(AM.LE.AMAX) GO TO 3
      AMAX=AM
      GO TO 2
    3 IF(AM.GE.AMIN) GO TO 2
      AMIN=AM
    2 CONTINUE
      WRITE(6,4)AMAX,AMIN
    4 FORMAT(2E15.5)
      STOP
      END
```

2.154

```
      DIMENSION A(20,20)
      READ(5,1)A
    1 FORMAT(10F8.3)
      AMAX=A(1,1)
      NMAX=1
      DO 2 I=2,20
      IF(AMAX-A(I,I))3,2,2
    3 AMAX=A(I,I)
      NMAX=I
    2 CONTINUE
      WRITE(6,5)(A(NMAX,J),J=1,20)
    5 FORMAT(10F10.3)
      STOP
      END
```

2.155

```
      DIMENSION X(40)
      READ(5,1)X
    1 FORMAT(16F5.2)
      XMAX=-1.E20
      DO 2 I=1,40
      IF(X(I))2,2,3
    3 IF(X(I).GT.XMAX)XMAX=X(I)
    2 CONTINUE
      WRITE(6,4)XMAX
    4 FORMAT(F6.2)
      STOP
      END
```

2.158

```
      READ(5,1)X
    1 FORMAT(F5.1)
      Y0=2.*(2.-SQRT(2.))
    2 Y1=1.5*Y0-0.5*X*Y0**3
      IF(ABS(Y1-Y0).LE.1.E-5) GO TO 3
      Y0=Y1
      GO TO 2
    3 WRITE(6,4)Y1,Y0
    4 FORMAT(2E13.5)
      STOP
      END
```

2.160

```
      X0=0.
      Y0=0.
    1 X2=X0**2
      Y2=Y0**2
      X1=X2+Y2+0.05
      Y1=X2-Y2+0.23
      IF(ABS(X1-X0).LE.1.E-4.AND.
     *ABS(Y1-Y0).LE.1.E-4) GO TO 2
      X0=X1
      Y0=Y1
      GO TO 1
    2 WRITE(6,3)X1,X0,Y1,Y0
    3 FORMAT(4E15.6)
      STOP
      END
```

2.164

```
      DIMENSION X(40),Z(40)
      READ(5,1)X
    1 FORMAT(20F4.1)
      DO 2 J=1,40
      Z(J)=1.
      DO 2 I=1,20
    2 Z(J)=Z(J)*(1.+1./(EXP(I)*X(J)))
      WRITE(6,3)Z
    3 FORMAT(10E12.4)
      STOP
      END
```

2.165

```
      DIMENSION A(20,20)
      READ(5,1)A
    1 FORMAT(10F8.3)
      S=0.
      DO 2 I=1,19
      K=I+1
      DO 2 J=K,20
    2 S=S+A(I,J)
      WRITE(6,3)S
    3 FORMAT(E15.5)
      STOP
      END
```

2.169

```
      DIMENSION X(20,20)
      READ(5,1)X
    1 FORMAT(16F5.2)
      SMIN=1.E30
      DO 2 I=1,20
      S=0.
      DO 3 J=1,20
    3 S=S*X(I,J)
      IF(S.GE.SMIN) GO TO 2
      SMIN=S
      IMIN=I
    2 CONTINUE
      WRITE(6,4)SMIN,IMIN
    4 FORMAT(E12.4,I4)
      STOP
      END
```

2.171

```
      DIMENSION B(20),C(20)
      READ(5,1)B
    1 FORMAT(10F8.3)
      DO 2 I=1,20
      X=-2.
      YMAX=-1.E30
      DO 3 J=1,41
      Y=2.*EXP(B(I)*X-5.*X*X)
      IF(Y.GT.YMAX) YMAX=Y
    3 X=X+0.1
    2 C(I)=YMAX
      WRITE(6,4)C
    4 FORMAT(10E12.4)
      STOP
      END
```

2.173

```
      DIMENSION A(10,20),M(20)
      READ(5,1)A
    1 FORMAT(20F4.1)
      DO 2 J=1,20
      M(J)=0
      DO 2 I=1,10
    2 IF(A(I,J).GT.0) M(J)=M(J)+1
      WRITE(6,3)M
    3 FORMAT(20I5)
      STOP
      END
```

2.176

```
      DIMENSION X(15,20)
      READ(5,1)X
    1 FORMAT(15F5.2)
      XMIN=X(1,1)
      DO 2 I=1,15
      DO 2 J=1,20
      IF(X(I,J)-XMIN)3,2,2
    3 XMIN=X(I,J)
      IMIN=I
      JMIN=J
    2 CONTINUE
      DO 4 I=1,15
    4 X(I,JMIN)=0
      DO 5 J=1,20
    5 X(IMIN,J)=0.
      WRITE(6,6)((X(I,J),J=1,20),
     *I=1,15)
    6 FORMAT(20F6.2)
      STOP
      END
```

2.177

```
      DIMENSION A(20)
      READ(5,1)A
    1 FORMAT(10F8.3)
      Z=0.
      DO 2 I=1,20
      P=1.
      B=0.
      DO 3 K=1,10
      P=P*(A(I)+B)/2.
    3 B=B+0.1
```

```
2  Z=Z+A(I)*P
   WRITE(6,4)Z
4  FORMAT(E12.4)
   STOP
   END
```

2.179

```
   DIMENSION A(10,20),B(20,10),
  *C(10,10)
   READ(5,1)N,M,L,((A(I,J),I=1,
  *N),J=1,M),((B(I,J),I=1,M),J=1,L)
1  FORMAT(3I2/(20F4.1))
   DO 2 I=1,N
   DO 2 K=1,L
   S=0.
   DO 3 J=1,M
3  S=S+A(I,J)*B(J,K)
2  C(I,K)=S
   DO 5 I=1,N
5  WRITE(6,4)(C(I,K),K=1,L)
4  FORMAT(10E12.4)
   STOP
   END
```

2.181

```
   DIMENSION A(15,25),B(15)
   READ(5,1)A,C
1  FORMAT(20F4.1)
   DO 2 I=1,15
   DO 3 J=1,20
   IF(A(I,J).GT.C) GO TO 4
3  CONTINUE
   B(I)=0.
   GO TO 2
4  B(I)=A(I,J)
2  CONTINUE
   WRITE(6,5)B
5  FORMAT(15F6.1)
   STOP
   END
```

2.185

```
   DIMENSION X(20,20)
   READ(5,1)X
1  FORMAT(16F5.2)
   DO 2 I=1,20
   XMIN=X(I,1)
   JMIN=1
   DO 3 J=2,20
   IF(XMIN-X(I,J))3,3,4
4  XMIN=X(I,J)
   JMIN=J
3  CONTINUE
   X(I,JMIN)=X(I,I)
2  X(I,I)=XMIN
   WRITE(6,4)((X(I,J),J=1,20),I=1,20)
4  FORMAT(20F6.2)
   STOP
   END
```

2.186

```
   READ(5,1)A,B,C,D
1  FORMAT(4F4.1)
   YMAX=-1.E30
   YMIN=1.E30
   X=-10.
   H=0.1
6  DO 2 I=1,201
   Y=A*X**3+B*X*X+C*X+D
   IF( YMAX.GE.Y ) GO TO 3
   YMAX=Y
   GO TO 2
3  IF(YMIN.LE.Y ) GO TO 4
   YMIN=Y
2  X=X+H
4  IF(H.LE.0.01) GO TO 5
   X=X-2*H
   H=0.01
   GO TO 6
5  WRITE(6,7)YMIN,YMAX
7  FORMAT(2E15.6)
   STOP
   END
```

2.198

```
   F(X,N)=COS(X)**(1./N)
   E(X,D)=EXP(X*X+D)
   READ(5,1)A,B,C
1  FORMAT(3F5.2)
   Z=E(A,-1.)*F(B,2)/(1-E(B,0.)+
  *(F(A+B,3.)+0.5)/(2*SQRT(E(C,1.))
   WRITE(6,2)Z
2  FORMAT(E12.4)
   STOP
   END
```

2.199

```
   F(T,Z)=T+(Z/T**2-T)/3.
   READ(5,1)X
1  FORMAT(F5.1)
   IF(X-1)2,3,3
2  YO=3.*X
   GO TO 4
3  YO=X/3.
4  Y1=F(YO,X)
   IF(ABS(YO-Y1).LT.1.E-5) GO TO 5
   YO=Y1
   GO TO 4
5  WRITE(6,6)YO,Y1
6  FORMAT(E15.7)
   STOP
   END
```

2.200

```
   INTEGER C
   READ(5,1)N,M
1  FORMAT(2I2)
   C=NFACT(N)/(NFACT(M)*NFACT(N-M))
   WRITE(6,2)C
2  FORMAT(I15)
   STOP
   END
   FUNCTION NFACT(N)
   IF(N)1,2,3
1  NFACT=0
   RETURN
2  NFACT=1
   RETURN
3  K=1
   DO 4 I=1,N
4  K=K*I
   NFACT=K
   RETURN
   END
```

2.204

```
   REAL MIN
   DIMENSION A(40),B(60)
   EXTERNAL SQRT,ABS,MIN
   READ(5,1)A,B
1  FORMAT(20F4.1)
   Z=(SUM(A,40,SQRT,MIN)+
  *SQRT(SUM(B,60,ABS,MIN)))/2.
   WRITE(6,2)Z
2  FORMAT(E15.5)
   STOP
   END
   FUNCTION SUM(X,N,F,F1)
   DIMENSION X(N)
   S=0.
   DO 1 I=1,N
1  S=S+F(X(I))/F1(X,I)
   SUM=S
   RETURN
   END
   REAL FUNCTION MIN(X,N)
   DIMENSION X(N)
   XMIN=X(1)
   DO 1 J=1,N
1  IF(X(J).LT.XMIN)XMIN=X(J)
   MIN=XMIN**2
   RETURN
   END
```

2.206

```
      READ(5,1)A,B,C
    1 FORMAT(3F4.1)
      CALL SQR(A,B,-1.5,X1,X2)
      CALL SQR(2.,-1.,C,Y1,Y2)
      U=(EXP(X1+Y1)-EXP(X2+Y2))
      WRITE(6,2)U
    2 FORMAT(E12.4)
      STOP
      END
      SUBROUTINE SQR(A,B,C,Z1,Z2)
      D=B*B-4.*A*C
      IF(D)1,2,2
    1 Z1=0.
      Z2=0.
      RETURN
    2 Z1=(-B+SQRT(D))/(2.*A)
      Z2=(-B-SQRT(D))/(2.*A)
      RETURN
      END
```

2.208

```
      INTEGER X,Y,A1,A2,A3
      READ(5,1)X,Y
    1 FORMAT(2I9)
      CALL EKS(X,Y,A1,A2)
    4 CALL EKS(A1-A2,A2,K1,K2)
      A3=K1-K2
      IF(A1-A3)2,3,2
    2 A1=A2
      A2=A3
      GO TO 4
    3 WRITE(6,5)A3
    5 FORMAT(I10)
      STOP
      END
      SUBROUTINE EKS(N,M,MAX,MIN)
      MAX=ABS(N)
      MIN=ABS(N)
      IF(M.GT.MAX) MAX=M
      IF(M.LT.MIN) MIN=M
      RETURN
      END
```

2.209

```
      DIMENSION A(21),B(40),C(41)
      READ(5,1)A,B,C
    1 FORMAT(10F8.3)
      CALL SUM(A,21,3,2,X1)
      CALL SUM(B,40,2,2,X2)
      CALL SUM(C,41,1,1,X3)
      Z=X1+X2+X3
      WRITE(6,2)Z
    2 FORMAT(E12.4)
      STOP
      END
      SUBROUTINE SUM(X,N,N0,NH,Z)
      DIMENSION X(N)
      Z=0.
      DO 1 I=N0,N,NH
    1 Z=Z+X(I)
      Z=Z/NFACT(N)
      RETURN
      END
      FUNCTION NFACT(N)
      K=1
      DO 1 I=1,N
    1 K=K*I
      NFACT=K
      RETURN
      END
```

2.210

```
      REAL MX,T1(200)
      COMMON N,MX,DX,T(200)
      EQUIVALENCE(T(1),T1(1))
      READ(5,1)T
    1 FORMAT(10F8.3)
      N=200
      CALL STAT
      WRITE(6,2)MX,DX
    2 FORMAT(2E14.6)
      DO 3 K=1,199
      TMIN=T(K)
      K1=K+1
      DO 4 I=K1,200
      IF(T(I)-TMIN)5,4,4
    5 TMIN=T(I)
      IMIN=I
    4 CONTINUE
      T(IMIN)=T(K)
    3 T(K)=TMIN
      DO 6 I=1,199
    6 T1(I)=T(I+1)-T(I)
      N=199
      CALL STAT
      WRITE(6,2)MX,DX
      STOP
      END
      SUBROUTINE STAT
      COMMON N,S,D,X(200)
      S=0.
      DO 1 I=1,N
    1 S=S+X(I)
      S=S/N
      D=1.
      DO 2 I=1,N
    2 D=D+(X(I)-S)**2
      D=D/N
      RETURN
      END
```

2.111

```
      DIMENSION A(40),B(50)
      READ(5,1)A,B
    1 FORMAT(16F5.1)
      CALL EKSTR(B,50,0,BS)
      CALL SUM(A,B,40,50,K,X,Y,&2)
      CALL EKSTR(A,40,K,AMAX)
      T=Y*AMAX/BS
      WRITE(6,3)T
    3 FORMAT(E14.6)
      GO TO 4
    2 CALL EKS(A,40,AMIN)
      R=X*AMIN/BS
      WRITE(6,3)R
    4 STOP
      END
      SUBROUTINE SUM(X,Y,N,M,K,SX,SY,*)
      DIMENSION X(N),Y(M)
      SX=0.
      SY=0.
      DO 1 I=1,N
    1 SX=SX+X(I)
      DO 2 I=1,M
    2 SY=SY+Y(I)
      IF(SX-SY)3,4,4
    3 K=1
      RETURN
    4 K=0
      RETURN 1
      END
      SUBROUTINE EKSTR(X,N,K,XM)
      DIMENSION X(N)
      XM=X(1)
      DO 1 I=2,N
    1 IF(X(I).GT.XM) XM=X(I)
      IF(K.EQ.0) GO TO 3
      RETURN
      ENTRY EKS(X,N,XM1)
      XM=0.
    3 XM1=X(1)
      DO 2 I=2,N
    2 IF(X(I).LT.XM1)XM1=X(I)
      XM=XM+XM1
      RETURN
      END
```

3.1

а) 1), 2), 3), 4), 6), 9)
б) 10), 11), 19), 21)

3.3

1) 0.250E2 2)0.100E7 3)0.343E11 4)0.529E-3

3.4

а) 1), 5), 6), 8), 9), 11)
б) 1) 5 5) 7 6) -10 8) -1,5 9) -0.50 11) 0

3.5

1), 2), 5)

3.6

1), 3), 5), 10), 11), 12)

3.10

K-DEC REAL FIXED(5,0); L-BIN REAL FLOAT(6);
E,F-DEC REAL FIXED(3); G,H-DEC REAL FLOAT(4)

3.11

BETA,GAMMA,MIN,MAX-DEC REAL FLOAT(6); SUMMA -DEC REAL FLOAT(6);
N,I-BIN REAL FIXED(15,0)

3.14

1) DCL(R,Q) REAL DEC FIXED(11,3); 2) DCL (X,Y,Z) REAL FLOAT(8);
3) DCL (I,J,K) REAL BIN FIXED(15,0); 4) DCL A(100) REAL FIXED(6,0);
5) B(10,10) REAL FIXED DEC(4,0);

3.15

ОПУЩЕННЫЕ ОПИСАТЕЛИ ПО УМОЛЧАНИЮ
1) DCL X FIXED(4); 2) DCL X FLOAT(3); 3) DCL X BIN FIXED(4,1);
4) DCL X FIXED(4,1); 5) DCL X BIN(3); 6) DCL X CHAR(5);
7) DCLXCHAR(4); 8) DCL X FIXED(5); 9) DCL X FLOAT(5);

3.16

1) DCL X(0:199) REAL DEC FIXED(8,2);
2) DCL X(-2:7,-2:7) REAL DEC FLOAT(6);
3) DCL X(0:9,-5:4,1:3) CHAR(16);

3.17

1) DCL A(10) FIXED(8); 2) DCL X(0:19) FIXED(8);
3) DCL D(-8:25) FIXED(8); 4) DCL A(3,4) FIXED(8);
5) DCL X(3,5,2) FIXED(8); 6) DCL X(0:2,0:3,1:2)FIXED(8);

3.19

1) X/(1+X*X/2/Y) 2) (A+B*X)/(C+D) 3) EXP(X)-SIN(X)**2
4) 1+ATAN(X/(1+SQRT(X))) 5) A*X**3-B*X*X+A*COS(ABS(X))**3
6) (SIN(X)**3-COS(X**3))/(2*X)-1E-3
7) (X**M-A**M)/SQRT(A*X) 8) EXP(-A*X)*SIN(OMEGA*T+FI)

3.20

1)5*SIN(X)+X**7*(1 +(X-Z/X)/(X+Z/X))*SIN(X)-1
2)3.14*R*R-V*V*H+(V*H+1)/V+3.14*R**3
3)(A+B)*SIN(A)+(A-B)*COS(B)+(A-B)/(SIN(A)+COS(B))
4) 1+X+X*X/2 +X**3/6 5) 1 +X/(1+X/(1+X)))
6) ((X**A)**B)**C 7) (X**Y/Y**X)**Z/Z/Y

3.23

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

3.24

1) '1'B 2) '0'B 3) '0'B 4) '1'B 5) '1'B

3.25

1) X**5*Y**3>0 2)B*B-4*A*C>=0 3) LOG(X(I,J))>0
4) LOG10(X)>0

3.26

1) '0100'B 2) '10001'B 3) '11110'B 4)'10111'B
5) '01100000'B 6)'0111011101000'B 7) 'ALFA␣PLUS␣␣'

3.27

1) '0'B 2) '1'B 3) '0'B 4) '1'B 5) '1'B

3.28

а) 1) C¬=0& Y>0 2)Y>0 & X-Y¬=0 3) X>=0& Y>0&LOG(Y)+X¬=0
4) B+C¬=0 & A+1/(B+C)¬=0 5) B¬=0 & C¬=0 & D¬=0
б) MOD(N,3)=0 б) MOD(K,5)=0&MOD(L,5)=0

3.29

1) 2, 1, 4, 3 2)1,2,3,5,4 3) 1,2,5,4,3,6
4) 1,3,2

3.30

X=0025.3 A=0000.1 B-ПРИСВАИВАНИЕ НЕ ПРОИЗВОДИТСЯ
C=-0018.2 D=2384,2

3.31

A=7.0200 B=00000025 C='ABCD␣' D='ПРОБЛЕ' F='10110'B

3.32

DCL X FIXED(3),Y FIXED(6,3),Z FLOAT(3);
-209,353,429;0.28E-7

```
3.33
1) A=-00.138     B=02.935    C=79.693    E=-001.0005    F-БЕЗ ПРИСВАИВАНИЯ
   D=198.0000    P='ТЕМПЫ РОСТА'    Q='ГОДОВОЙ ДОХОД'
2) A=-1.38000E+02    B=2.93500E+03    E=1.79693E+05    F=-1.00050E+04
   C-ПРИСВАИВАНИЕ НЕ ПРОИЗВОДИТСЯ    D=2.19800E+03

3.36
СУММА=    -318.25          ПРОИЗВЕДЕНИЕ= 1.000E 04

3.37
-1.38000E+02                2.93500E+03
1.79693E+05                 2.19800E+03              -1.00050E 04

3.38
1)  29.83                   БАЛАНС= 4.3E-07
3) ПЕРИОД=    999.110          ФУНКЦИЯ=      2.897

3.41
X=456      Y=83      Z=29      A=82.1      B=-1.3
   82.1                 -1.3

3.43
A=1    B=345    C=12

3.44
1) PUT EDIT(Z)(F(5));    2) PUT EDIT(Z)(F(7,2));
3) PUT EDIT(Z)(E(9,2));      4) PUT EDIT ('Z=',Z)(A,F(5));

3.45
1) A=1234    B=5678    2) A=1.23  B=45.67    3) A=501.2    B=39.1
4) A=1234.5    B=6.7E-2    5) A=2 E9    B=0.749

3.46
1) РАСХОД    РАСХОД    3) АРГУМЕНТ          ФУНКЦИЯ
                          0.001             -8.290

3.47
1) PUT EDIT('ТАБУЛИРОВАНИЕ ФУНКЦИИ','X','Y')(A,COL(4),A,COL(16),A);
3) PUT EDIT('КОРНИ УРАВНЕНИЯ','X1=',X1,'X2=',X2)
          (SKIP,A,2(SKIP,X(3),A,F(7,4)));

3.48
F =0899.35    B=-0029.03    C-НЕ ИЗМЕНЯЕТСЯ    F=1.500E-06    I='NAME  '

3.50
X=005.00      Y=-006.00      Z=007.94    W='ABCD    '

3.51
1) X= 1.00000E+01      Y=-8.00000E+00      Z=0.00000E+00
2)  x=    -847      Y=  -1000    W='АВАНС'

3.52
M=7.8,N=7.8,A=1E-06,Q=-8E04,X=1,Y=1,Z=1;

3.57
DCL(S,Q)(100)FIXED(6,2); GET EDIT (S,Q)(SKIP,10F(6,2));

3.58
DCL(X,Y,Z)FIXED(4),A(50)FIXED(7,3);
GET EDIT(X,Y,Z,A)(SKIP,4F(4),10(SKIP,5F(7,3)));

3.60
DCL X(80)FIXED(6);  GET EDIT(K) (SKIP,F(2));
GET EDIT ((X(I)DO I=1 TO K))(SKIP,8F(6));

3.61
 8   8.0 -4.0 -2.00   0   2.0   4.0   6.00  8 10.0

3.62
DCL (X,Y,Z )(20)FIXED; DO I=1 TO 20;
GET EDIT((X(I),Y(I),Z(I))(SKIP,3F(6,0)); END;

3.63
4) DO I=1 TO 4; DO J=1 TO 25;  GET EDIT((B(I-1)*25+J),C(I,J))
     (2F(7,3));END;END;GET EDIT(D)(SKIP,5F(7,3));

3.65
DCL(X,Y)(25)FLOAT;  PUT EDIT('МАССИВ X',(8)'-',X)
          (SKIP,A,SKIP(0),A, 5(SKIP,5E(10,3),X(5)));
PUT EDIT('МАССИВ Y',(8)'-',Y)(SKIP,A,SKIP(0),A, 5(SKIP,
          5E(10,3),X(5))));

3.66
DCL(X,F)(20)FLOAT; PUT EDIT('АРГУМЕНТ','ФУНКЦИЯ',(X(I,F(I)DO I=1
TO 20))(SKIP,A,X(20),A,(SKIP,F(5,2),X(24),E(12,3)));

3.67
DCL(A,B,C)(8) FLOAT; PUT EDIT((I,A(I),B(I),C(I)  DO I=1 TO 8))
                          (SKIP,F(1),3E(20,5));

3.69
DCL Z(5,10)FIXED(8,3);DO I=1 TO 5;  PUT EDIT(I,(Z(I,J)DO J=1 TO 10))
(SKIP,F(1),X(5),10(F(8,3),X(3)));END;

3.70
DCL Z(5,10)FIXED(8,3), NUMER(10)FIXED(2);PUT EDIT(NUMER)
(SKIP,X(7),F(2),9(X(10),F(2)));DO I=1 TO 5; PUT EDIT (I,(Z(I,J)
DO J=1 TO 10)) (SKIP,F(1),X(3),10(X(3),F(8,3)));END;
```

3.81

```
W3:PROC OPTIONS(MAIN);
   DCL(X,Y,M)(3)FIXED(5,2),
   M1 FIXED(7,2);
   GET EDIT(X(I),Y(I),M(I)
   DO I=1 TO 3)(SKIP,3F(5,2));
   M1=M(1)+M(2)+M(3);
   XC=(M(1)*X(1)+M(2)*X(2)*
   M(3)*X(3))/M1;
   YC=(M(1)*Y(1)+M(2)*Y(2)+
   M(3)*Y(3))/M1;
   PUT EDIT(XC,YC)(SKIP,
   2(F(5,2),X(10)));
   END W3;
```

3.86

```
W8:PROC OPTIONS(MAIN);
   DCL(A,S,X)FLOAT;
   GET EDIT(A,S)(F(5,2));
   X=-A/2+SQRT(A*A/4+2*S);
   PUT EDIT('ВЫСОТА ТРЕУГОЛЬНИКА=',
   X)(COL(5),A,F(7,2));
   END W8;
```

3.90

```
W11:PROC OPTIONS(MAIN);
   DCL(A,B,C)FLOAT;
   GET DATA(A,B,C);
   D=B*B-4*A*C;
   Z=2*A;E=-B/Z;
   F=SQRT(ABS(D))/Z;
   IF D>=0 THEN DO;
   X1=E+F;X2=E-F;
   PUT EDIT('КОРНИ
   ДЕЙСТВИТЕЛЬНЫЕ',X1,X2)
   (SKIP,A,COL(4),2E(12,3));END;
   ELSE PUT EDIT('КОРНИ МНИМЫЕ',
   E,F)(SKIP,A,COL(4),2E(12,3));
   END W11;
```

3.92 а)

```
W13:PROC OPTIONS(MAIN);
   DCL(X,Y)FIXED(6,3);
   GET EDIT(X)(F(6,3));
   IF X<=-1 THEN Y=0;
   ELSE Y=1-ABS(X);
   PUT EDIT('Y=',Y)(COL(1),A,F(6,3));
   END W13;
```

3.93

```
W14:PROC OPTIONS(MAIN);
   DCL C BIT(1),
   X FIXED(5,2);
   GET DATA(X);
   IF X>0 THEN C='1'B;ELSE C='0'B;
   PUT DATA(' ',C);END W14;
```

3.94

```
W15:PROC OPTIONS(MAIN);
   DCL(A,B)FIXED(5,1),N FIXED(1);
   GET LIST(A,B);
   IFA>B THEN DO;N=1;Z=A*A;END;
   ELSE DO;N=2;Z=B*B;END;
   PUT LIST(Z,N);
   END W15;
```

3.95

```
W16:PROC OPTIONS(MAIN);
   DCL(R,X0,Y0)FIXED(5,2);
   GET EDIT(R,X0,Y0)
   (F(5,2),SKIP,2F(5,2));
   IF SQRT(X0*X0+Y0*Y0)<R
   THEN N=1;ELSE N=0;
   PUT EDIT('N=',N)(COL(5),A,F(1));
   END W16;
```

3.96

```
W17:PROC OPTIONS(MAIN);
   DCL(X,Z)FLOAT;
   GET DATA(X);
   IF X>=1  THEN Z=LOG(X);
   ELSE IF X<=-1
   THEN Z=EXP(X);ELSE Z=1 ;
   PUT DATA(Z);
   END W17;
```

3.97 б)

```
W18:PROC OPTIONS(MAIN);
   DCL(X,Y)FIXED(5,2),N FIXED(1);
   GET EDIT(X,Y)(2F(5,2));
   IF X>0& Y<0 THEN N=4;
   ELSE IF X<0 & Y<0 THEN N=3;
   ELSE IF X<0& Y>0 THEN N=2;
   ELSE N=1;
   PUT EDIT('НОМЕР КВАДРАТА='
   ,N)(SKIP,X(5),A,F(1));
   END W18;
```

3.98

```
W19:PROC OPTIONS(MAIN);
   DCL X FIXED(3,1),NX FIXED(1);
   GET DATA (X);
   IF X+0.5<4.5
   THEN NX=X+0.5;ELSE NX=5;
   PUT DATA(NX);
   END W19;
```

3.100

```
W20:PROC OPTIONS(MAIN);
   DCL X FIXED(3);
   GET LIST(X);
   IF DIVIDE(X,3,3,0)*3=X
   THEN PUT DATA(X);
   ELSE PUT EDIT('НЕТ')(A);
   END W20;
```

3.106

```
W25:PROC OPTIONS(MAIN);
   DCL A(25)FIXED(5,2);
   GET LIST(A);
   DO I=1 TO25;
   IFA(I)<0 THEN A(I)=0;
   END;
   PUT LIST(A);
   END W25;
```

3.107

```
W26: PROC OPTIONS(MAIN);
   DCL X(20)FIXED(4,1);
   GET LIST(X);
   DO I=1 TO 20; X(I)=0;END;
   PUT LIST(X);
   END W26;
```

3.108 а)

```
W27:PROC OPTIONS(MAIN);
   DCL(X,A)FIXED(6,2);
   GET DATA(X,A);
   K=1;
M1:Z=X**K/K**2;
   IF Z>A THEN DO;
   PUT EDIT(Z)(SKIP,E(10,3));
   K=K+1;GO TO M1;END;
   END W27;
```

3.109

```
W28:PROC OPTIONS(MAIN);
   DCL(X,A)FIXED(6,2),
   K FIXED(3);
   GET LIST(X,A);K=1;
M2:IF X**K/K**2>A THEN
   DO;K=K+1;GO TO M2;END;
   ELSE PUT DATA(K);
   END W28;
```

3.111 б)

```
W30:PROC OPTIONS(MAIN);
   DCL A(50)FIXED(6,3);
   GET LIST(A);
L: DO I=1 TO 50 WHILE(A(I)<0);
   END L;
   PUT DATA(I,A(I));
   END W30;
```

3.112

```
W31:PROC OPTIONS(MAIN);
   DCL Y(100)FIXED(6,1);
   GET LIST(Y);
L: DO I=1 TO 100;
   IF Y(I)>0&Y(I)<1
   THEN PUT LIST(I);
   END L;
   END W31;
```

3.114

```
W33:PROC OPTIONS(MAIN);
   DCL(X,Y)FIXED(5,2);X=2;
M1:Y=X-ATAN(X)-3.14159;
   IF Y<0 THEN DO;X=X+0.1;
   IF X>5 THEN DO;PUT LIST
   ('КОРЕНЬ НЕ НАЙДЕН')SKIP;GO TO M2;END;
   ELSE GO TO M1;END;
   ELSE PUT DATA(X) SKIP;
M2:END W133;
```

3.115

```
W34:PROC OPTIONS(MAIN);
   DCL(X,Y)(100)FIXED(5,1);
   GET EDIT((X(I),Y(I)DO I=1 TO 100),
   R)(SKIP,10F(5,1));
   R1=R*R;
L: DO I=1 TO100;
   IF X(I)**2+Y(I)**2<=R1
   THEN PUT LIST(I);END L;
   END W34;
```

3.116

```
W35:PROC OPTIONS(MAIN);
    DCL(A,B0,BM)FLOAT;
    GET DATA(A,B0,BM);
    X=B0;Y=X-EXP(-(A*X)**2/2);
    IF Y<0 THEN N=-1;ELSE N=1;
T1: X=X+0.2;IF X>BM
    THEN DO;PUT LIST('КОРЕНЬ НЕ НАЙДЕН')
    ;GOTOM1;END;
    Y=Y-EXP(-(A*X)**2/2);
    IF N*Y>0 THEN GO TOT1;
    ELSE PUT DATA(X);
M1: END W35;
```

3.118 б)

```
W37:PROC OPTIONS(MAIN);
    DCL(R,A4)FLOAT(5);
    GET DATA(R,A4);
    R1=R*R;DO N=1 TO5;
    A2N=SQRT(2*R1-2*R*
    SQRT(R1-A4**2/4));
    A4=A2N;END;
    PUT DATA(A2N);
    END W37;
```

3.119

```
W38:PROC OPTIONS(MAIN);
    DCL X(100,100)FIXED(5,2);
    GET EDIT(N,((X(I,J)DO J=1 TON)
    DO I=1 TON))
    (SKIP,F(3),(COL(1),10F(5,2)));
L:  DO I=1 TO N;
    IF X(I,I)>0 THEN
    PUT LIST(X(I,I));ENDL;END W38;
```

3.121

```
W40:PROC OPTIONS(MAIN);
    DCL N FIXED(2);
    GET DATA(N);
    A,C=0.112;B=SQRT(1-A*A);
    DO I=2 TON;D=SQRT(1-C*C);
    C=C*A-D*B;END;
    PUT DATA(C);
    END W40;
```

3.122

```
W41:PROC OPTIONS(MAIN);
    DCL N FIXED(9,0);
    GET DATA(N);
    K=0;M2:N=N/10;K=K+1;
    IF N¬=0 THEN GO TO M2;
    PUT DATA(K);
    END;
```

3.123

```
W42:PROC OPTIONS(MAIN);
    DCL X(50)FIXED(4);
    GET LIST(N,(X(I)DO I=1 TON));
A:  DO I=1 TO N;
    IF DIVIDE(X(I),3,4,0)*3=
    X(I) THEN PUT LIST(X(I));
    ENDA:END W42;
```

3.126

```
W45:PROC OPTIONS(MAIN);
    DCL C(11) FIXED(6,2);
    GET LIST(C);
    A=0;B=1;
    DO I=1 TO 11;
    Z=(A+B+C(I))/I;
    PUT EDIT(A,B,C(I),Z)
    (SKIP,4F(12,2));
    A=A+0.1;B=B+0.2;
    END;END W45;
```

3.128

```
W47:  PROC OPTIONS(MAIN);
      DCL(Y0,H,Z)FIXED(5,1);
      GET DATA(Y0,H);Y=Y0;
   K: DO X=1 TO 2 BY 0.1;
      Z=X*SQRT(X+Y);
      PUT LIST(X,' ',Y,' ',Z);
        Y=Y+H; END K; END W47;
```

3.129

```
W48:  PROC OPTIONS(MAIN);
      GET DATA(X,H);
      DO N=3 TO 41 BY2;
      X=X+H; Y=X/N;
      PUT LIST(  Y);END;
      END W48;
```

3.130

```
W49:PROC OPTIONS(MAIN);
    DCL(X0,DX)FIXED(5,2),
    (N0,DN,M)FIXED(2);
    GET LIST(X0,DX,N0,DN,M);
    K=2*DN;X=X0+DX;
    DO N=N0+3*DN TO M BYK;
    Y=X/N;PUT LIST(Y);
    X=X+2*DN;END;END W49;
```

3.132

```
W51:PROC OPTIONS(MAIN);
    DCL Z(21)FLOAT;
    GET DATA(A,B,C);
    X=0;DO I=1 TO21;
    Z(I)=A*EXP(B*X-C*X*X);
    X=X+0.1;END;
    PUT EDIT(Z)(SKIP,8E(12,3));
    END W51;
```

3.137

```
W53:PROC OPTIONS(MAIN);
    DCL(X,Y)(20)FIXED(6,1);
    GET EDIT(X)(SKIP,10F(6,2));
A:  DO I=1 TO 20;
    Y(I)=0;
    IF X(I)>0 THEN Y(I)=X(I);
    ELSE IF X(I)<0 THEN Y(I)=-1;
    END A;PUT DATA(Y);
    END W53;
```

3.136

```
W55:PROC OPTIONS(MAIN);
    DCL(X,Y)(100)FIXED(4,1);
    GET LIST(X);K=0;
L:  DO I=1 TO 100;
    IF X(I)>0 THEN DO;K=K+1;
    Y(K)=X(I);END;END L;
    PUT EDIT((Y(I)DO I=1 TO K))
    (COL(1),10F(9,1));END W55;
```

3.138 б)

```
W57:PROC OPTIONS(MAIN);
    DCL(X(100),Y(10))FIXED(5,2);
    GET LIST(X);K=0;
A:  DO I=1 TO 100 WHILE(K< 10);
    IF X(I)>0 THEN DO;
    K=K+1;Y(K)=X(I);END;
    END A;
    PUT LIST((Y(I)DO I=1 TO K));
    END W57;
```

3.139

```
W58:PROC OPTIONS(MAIN);
    DCL Z(50);
    GET DATA(N,X0,XM,H);K=0;
  L: DO X=X0 BY H TO XM;
    Y=N*SIN(X)-COS(N*X);
    IF X>=0& X<=1 THEN DO;
    K=K+1;Z(K)=Y;END;
      END L;
    PUT LIST((Z(I)DO I=1 TO K));
    END W58;
```

3.141

```
W60:PROC OPTIONS(MAIN);
    DCL(A(100),(B,C)(50))FIXED(8,3);
    GET EDIT(A)(SKIP,10F(8,3));
    DO I=1 TO 50;
    B(I)=A(2*I);C(I)=A(2*I-1);
    END;
    PUT LIST(B,C)SKIP(2);
    END W60;
```

3.142

```
W61:PROC OPTIONS(MAIN);
    DCL Z(101)FLOAT;
    GET DATA(A,B,C);K=0;
  M: DO X=0 TO 10 BY 0.1;
    Y=-X**3+A*X*X+B*X+C;
    IF Y <0 THEN GO TO M1;
    K=K+1;Z(K)=Y;
      END M;
M1: PUT LIST((Z(I)DO I=1 TO K));
    END W61;
```

3.143

```
W62:PROC OPTIONS(MAIN);
    DCL(X,Y,Z)(60)FIXED(6,2);
    GET LIST(X);
    N=0;K=0;
A:  DO I=1 TO 60;
    IF X(I)>0 THEN DO;
    N=N+1;Z(N)=X(I);END;
    ELSE DO;K=K+1;Y(K)=X(I);
    END;END A;
```

```
      PUT EDIT('МАССИВ Z',
      (Z(I)DO I=1 TO N))
      (SKIP,A,4(COL(1),15(F(6,2),X(2))));
      PUT EDIT ('МАССИВ Y',
      (Y(I) DO I=1 TO K))'
      (SKIP,A,4(COL(1),15(F(6,2),X(2))));
      END;
```

3.153

```
W72: PROC OPTIONS(MAIN);
      GET DATA (X); S=1 ; Y=1 ;
      DO I=1 TO 19;
      Y=Y*X;S=S+Y/I;
      END;
      PUT DATA(S);
      END W72;
```

3.154

```
W73:PROC OPTIONS(MAIN);
      DCL C(30) FLOAT;
      GET LIST(C);
      S=0 ;N=0;
   L: DO I=1 TO 30;
      IF C(I)<0THEN DO;
      S=S+C(I);N=N+1;END;
      END L;S=S/N;
      PUT DATA (S);
      END W73;
```

3.155

```
W74:PROC OPTIONS(MAIN);
      DCL A(80) FIXED(5,1);
      GET LIST(A);S=0 ;N=0;
L:    DO I=1 TO 80;
      IF A(I)>=1 & A(I)<=2
      THEN DO;S=S+A(I);
      N=N+1; END;END L;
      IF N=0 THEN PUT EDIT('ЭЛЕМЕНТОВ
      НЕТ')(SKIP,A);
      ELSE PUT EDIT(S/N)
      (SKIP,F(8,2));END W74;
```

3.157

```
W76:PROC OPTIONS(MAIN);
      DCL Z FIXED(15),
      N FIXED(2);
      GET DATA(N);
      IF N<0 THEN DO;
      Z=0;GOTO M1;END;
      ELSE IF N=0 THEN DO;
      Z=1;GOTO M1;END;
      ELSE Z=1;
      DO I=1 TO N;
L:    Z=Z*Z*I;END L;
      PUT DATA (Z);
M1:   END W76;
```

3.164

```
W83:PROC OPTIONS(MAIN);
      DCL Y(25) FLOAT;
      GET EDIT(Y,A) (SKIP,8F(6,2));
      Z=1 ;N=0;
      DO I=1 TO 25;
      IF(Y(I)> A THEN DO;
      Z=Z*Y(I);N=N+1;END;
      Z=Z**(1 /N);PUT DATA(Z);
      END W83;
```

3.165

```
W84:PROC OPTIONS(MAIN);
      DCL A(40) FIXED(6,2);
      GET LIST(A);S=1;N=0;
      DO I=2 TO 40 BY2;
L:    IF A(I)>0 THEN DO;S=S*A(I);
      N=N+1; END;END L;
      IF N=0 THEN PUT EDIT
      ('ЭЛЕМЕНТОВ НЕТ S=0')(SKIP(2),A);
      ELSE DO; S=S**(1 /N);PUT DATA(S);
      END;END W84;
```

3.166

```
W85:PROC OPTIONS(MAIN);
      GET DATA (X,N);
      N1=1;A:DO I=1 TON;
      N1=N1*I;ENDA;
      S=(X/2)**N/N1;Y=S;
B   :DO K=1 TO 100;
      Y=Y*(-X*X)/(K*(K+N))/4;
      S=S+Y;END B;
      PUT EDIT(S)(SKIP(2),COL(8),F(8,2));
      END W85;
```

3.169

```
W88:PROC OPTIONS(MAIN);
      DCL (A(75),S)FIXED(8,0);
      GET LIST(A);S=0;
      DO I=1 TO 75; S=S+A(I);END;
      IF MOD(S,2)=0 THEN
      PUT EDIT('ДА')(SKIP,A);
      ELSE PUT EDIT('НЕТ')
      (SKIP,A);END W88;
```

3.170

```
W89:PROC OPTIONS(MAIN);
      DCL X(100)FIXED(6,2),
      NX FIXED(5,0);
      GET LIST(X);M=0;
      DO I=1 TO 100; NX=X(I)+0.5;
      IF NX=1 THEN M=M+1;
      END;
      PUT DATA(M);
      END W89;
```

3.171

```
W90:PROC OPTIONS(MAIN);
      DCL(S1,S2)FIXED(10);
      GET DATA(N);S1=0;
      S2=0;
L:    DO I=1 TON BY2;
      S1=S1+I;
      IF I=N THEN GOTO M2;ELSE
      S2=S2+I+1;END L;
M2:   PUT DATA(S1,S2);
      END W90;
```

3.176

```
W95:PROC OPTIONS(MAIN);
      DCL(X,EPS)FLOAT;
      GET LIST(X,EPS);
      Z=0;N=1;
M8:   Y=COS(N*X)/N**2;
      Z=Z+Y;N=N+1;
      IF Y>EPS THEN GOTO M8;
      PUT EDIT(Z) (SKIP,F(10,6));
      END W95;
```

3.178

```
W97:PROC OPTIONS(MAIN);
      GET DATA (X);Y=0;Z=1;
      DO N=1 WHILE(Z>1E-4);
      Z=COS(2*N*X)/(2*N-1)/(2*N+1);
      Y=Y+Z;END;
      PUT EDIT(Y)(SKIP,F(6,4));
      END W97;
```

3.179

```
W98:PROC OPTIONS(MAIN);
      DCL N FIXED(2,0);
      GET LIST(X,N);
      Y=X**N;Z=Y;
      DO I=2 WHILE(Z>1E-6);
      Z=Z*(I-1)/I/X; Y=Y+Z;
      END;
      PUT LIST(Y);
      END W98;
```

3.183

```
W102:PROC OPTIONS(MAIN);
      GET DATA(X);Z=1;
      DO N=2 TO9;Z=Z*X+N;END;
      PUT DATA(Z);END W102;
```

3.184

```
W103:PROC OPTIONS(MAIN);
      GET LIST(X);S=1;
      DO I=1 TO 8;S=S*X*I/(9-I)+1;
      END;PUT LIST(S);
      END W103;
```

3.185

```
W104:PROC OPTIONS(MAIN);
      GET DATA(X);Z=1;
      DO N=1 TO 12;Z=Z*X/(13-N)+1;END;
      PUT DATA(Z);END W104;
```

3.192

```
W111:PROC OPTIONS(MAIN);
      GET DATA(A,B,C,D);
      YMAX=-1E18;YMIN=1E18;
CIKL:DO X=-10 TO10 BY0.2;
      Y=A*X**3+B*X**X+C*X+D;
      IF Y>YMAX THEN YMAX=Y;
      ELSE IF Y<YMIN THEN YMIN=Y;
      ELSE GOTO R2; END CIKL;
R2:   PUT DATA(YMAX,YMIN);
      END W111;
```

3.194

```
W113:PROC OPTIONS(MAIN);
     GET LIST(A,B,C,XO,XM,H);
     Y=A*EXP(A*XO)-B*EXP(-C*XO);
     YM=A*EXP(A*(XO+H))-B*EXP(-C*(XO+H));
     N=1; IF Y<YM THEN N=-1;
 L: DO X=XO+2*H TO XM BY H;
     Y=A*EXP(A*X)-B*EXP(-C*X);
     IF N*Y┐<N*YM THEN GOTO R1;
    YM=Y;END L;
 R1: PUT DATA(YM,N);
     END W113;
```

3.195

```
W114:PROC OPTIONS(MAIN);
     DCL (X(50),XMAX,XMIN)FIXED(6,2);
     GET LIST(X);
     XMAX=X(1);XMIN=X(1);
     NMAX=1;NMIN=1;
L:   DO I=2 TO 50;
     IF X(I)>XMAX THEN DO;
     XMAX=X(I);NMAX=I;END;
     IF X(I)<XMIN THEN DO;
     XMIN=X(I);NMIN=I;END;
     END L;
     X(NMAX)=1; X(NMIN)=-1;
     PUT EDIT(X) (SKIP,10(F(6,2),X(4)));
     END;
```

3.196

```
W116:PROC OPTIONS(MAIN);
     DCL A(80)FIXED(6,2),
     (AR,AMAX,AMIN)FIXED(7,2);
     GET LIST(A);
     AMAX=ABS(A(1)-A(2));AMIN=AMAX;
CIKL:DO I=2 TO 79;
     AR=ABS(A(I)-A(I+1));
     IF AMAX<AR THEN AMAX=AR;ELSE
     IF AMIN<AR THEN AMIN=AR;
     END CIKL;
     PUT DATA(AMAX,AMIN);
     END W116;
```

3.199

```
W118:PROC OPTIONS(MAIN);
     DCL(A(20,20),AMAX,AMIN)FIXED(5,2);
     GET LIST(((A(I,J)DO J=1 TO 20)
     DO I=1 TO 20));
     AMAX=A(1,1);NMAX=1;
L:   DO I=2 TO 20;
     IF AMAX<A(I,I) THEN DO;
     AMAX=A(I,I);NMAX=I;END;
     END L;
     PUT EDIT((A(NMAX,J)DO J=1 TO 20)
     )(SKIP,10F(10,2));
     END W118;
```

3.200

```
W119:PROC OPTIONS(MAIN);
     DCL(X(40),AMIX)FIXED(4,1);
     GET EDIT(X)(SKIP,10F(8,1));
     AMIX= 1E19;
     DO I=1 TO40;
     IF X(I)>0 & X(I)<AMIX
     THEN AMIX=X(I);END;
     PUT DATA(AMIX)SKIP(2);
     END W119;
```

3.203

```
W122:PROC OPTIONS(MAIN);
     GET DATA(X);
     YO=2*(2-SQRT(2));
T2:  Y1=1.5*YO-0.5*X*YO**3;
     IF ABS(Y1-YO)< 1E-5
     THEN PUT DATA(Y1, 0);
     ELSE DO;YO=Y1;GO TO T2;END;
     END W122;
```

3.205

```
W124:PROC OPTIONS(MAIN);
     YO=0 ; XO=0 ;
T1:  X1=XO**2+YO**2+0.05;
     Y1=XO**2-YO**2+0.23;
     IF ABS(X1-XO)<1E-4 &
     ABS(Y1-YO)<1E-4 THEN GO TO R1;
     ELSE DO;XO=X1;YO=Y1;
     GO TO T1;END;
R1:  PUT EDIT(X1,XO,Y1,YO)
     (SKIP(2),2F(12,6));
     END W124;
```

3.209

```
W128:PROC OPTIONS(MAIN);
     DCL(X,Z)(40)FLOAT;
     GET EDIT(X)(SKIP,10F(6,2));
B:   DO J=1 TO 40;
     Z(J)=1 ;
A:   DO I=1 TO 20;
     Z(J)=Z(J)*(1+1/(EXP(I)+X(J)));
     END A;END B;
     PUT EDIT(Z)(SKIP,10E(12,3));
     END W128;
```

3.210

```
W129:PROC OPTIONS(MAIN);
     DCL A(20,20)FIXED(3,1);
     GET EDIT(A)(SKIP,20F(3,1));
     S=0;
     DO I=1 TO 19;
     DO J=I+1 TO 20;
     S=S+A(I,J); END;END;
     PUT EDIT('CYMMA=',S)(SKIP,A,F(8,2));
     END W129;
```

3.214

```
W133: PROC OPTIONS(MAIN);
     DCL(X(20,20),S)FIXED(8,3);
     GET EDIT X(SKIP,16F(5,3));
     SMIN=1E19;
L:   DOI=1 TO 20; S=0;
 M: DO J=1 TO 20; S=S+X(I,J);
    END M; IF S<SMIN THEN DO;
     SMIN=S;IMIN=I;END;
     END L;
     PUT EDIT(SMIN,IMIN)
     (COL(5),F(6,3),X(10),F(2));
     END W133;
```

3.216

```
W135:PROC OPTIONS(MAIN);
     DCL (B,C)(20)FLOAT;
     GET LIST(B);
L:   DO I=1 TO 20;
     YMAX=-1E19;
 M: DO X=-2 TO 2 BY 0.1;
     Y=2*EXP(B(I)*X-5*X*X);
     IF Y>YMAX THEN YMAX=Y;
    END M; C(I)=YMAX;
     END L;
     PUT EDIT(C)(SKIP,10 E(12,4));
     END W135;
```

3.217

```
W136:PROC OPTIONS(MAIN);
     DCL A(10,25)FIXED(4,2);
     GET LIST(A);N=0;M=0;
L:   DO I=1 TO 10;
 M2:DO J=1 TO 25;
     IF A(I,J)>0 THEN M=M+1;
     IF A(I,J)<0 THEN N=N+1;
    END M2;END L;
     PUT DATA (N,M);
     END W136;
```

3.218

```
W137:PROC OPTIONS(MAIN);
     DCL A(10,20)FIXED(5,1),
     M(20) FIXED(3);
     GET EDIT(A)(SKIP,15F(5,1));
 M8:DO J=1 TO 20; M(J)=0;
L:   DO I=1 TO 10;
IF A(I,J)>0 THEN M(J)=M(J)+1;
    END L; END M8;
     PUT EDIT(M) (SKIP,20F(6));
     END W137;
```

3.221

```
W140:PROC OPTIONS(MAIN);
     DCL(X(15,20),XMIN)FIXED(5,2);
     GET LIST(X);
     XMIN=X(1,1);IMIN=1;JMIN=1;
     L:DO I=1 TO 15;
     U:DO J=1 TO 20;
      IF X(I,J)<XMIN THEN DO;
     XMIN=X(I,J);IMIN=I;JMIN=J;
     END;END U;END L;
     DO J=1 TO 20;X(IMIN,J)=0;END;
     DO I=1 TO 15;X(I,JMIN)=0;END;
     PUT EDIT(((X(I,J)DO J=1 TO 20)
     DO I=1 TO 15))(SKIP,20F(6,2));
     END W140;
```

3.222

```
W141:PROC OPTIONS(MAIN);
     DCL A(20) FIXED(6,2);
     GET LIST(A);Z=0;
     L:DO I=1 TO 20;P=1;B=0;
     R:DO K=1 TO 10;
     P=P*(A(I)+B)/2;B=B+0.1;END R;
     Z=Z+A(I)*P; END L;
     PUT DATA(Z);
     END W141;
```

3.224

```
W143:PROC OPTIONS(MAIN);
     DCL(A(10,20),B(20,10))FIXED(4,1),
     C(10,10) FIXED(8,2),
     (N,M,L) FIXED(2);
     GET EDIT(N,M,L)(SKIP,3F(2));
     GET EDIT(((A(I,J) DO I=1 TO N)
     DO J=1 TOM),((B(I,J) DO I=1 TOM)
      DO J=1 TO L))(SKIP,20F(4,1));
     T:DO I=1 TO N;
     R:DO K=1 TO L;S=0;
     U:DO J=1 TOM;
       S=S+A(I,J)*B(J,K);
     END U;C(I,K)=S;
     END R;END T;
     DO I=1 TO N;
     PUT EDIT((C(I,K) DO K=1
     TO L))(SKIP,10F(12,2));
     END;END W143;
```

3.225

```
W144:PROC OPTIONS(MAIN);
     DCL A(100) FIXED(5,3),
     (MX,DX) FLOAT;
     DO J=1 TO3;
     GET LIST(A);MX,DX=0;
     DO I=1 TO 100;
     MX=MX+A(I);END;
     MX=MX/100;
     DO I=1 TO 100;
     DX=DX+(A(I)-MX)**2;END;
     DX=DX/100;
     PUT DATA(MX,DX);END;
     END W144;
```

3.226

```
W145:PROC OPTIONS(MAIN);
     DCL(A(15,25),B(15))FIXED(4,2);
     GET LIST(C,A);
     L:DO I=1 TO 15;
 M:  DO J=1 TO 25;
     IF A(I,J)>C THEN DO;
     B(I)=A(I,J);GOTO R8;END;
     END M; B(I)=0;
     R8:END L;
     PUT EDIT(B)(SKIP,15F(7,2));
     END W145;
```

3.230

```
W149:PROC OPTIONS(MAIN);
     DCL(X(20,20),XMIN)FIXED(5,2);
     GET LIST(X);
     L:DO I=1 TO 20;
     XMIN=X(I,1);JMIN=1;
 M:  DO J=2 TO 20;
     IF X(I,J)<XMIN THEN DO;
     XMIN=X(I,J);JMIN=J;END;
     END M;
     X(I,JMIN)=X(I,I);X(I,I)=XMIN;
     END L;PUT EDIT(X)(SKIP,20F(6,2));
     END W149;
```

3.231

```
W150:PROC OPTIONS(MAIN);
     GET EDIT(A,B,C,D)      (4F(5,2));
     YMAX=-1E19;YMIN=1E19;H=0.1;X0=-10;
  K: DO X= X0 TO 10 BY H;
     Y=A*X**3+B*X*X+C*X+D;
     IF Y>YMAX THEN YMAX=Y;
        ELSE IF Y <YMIN THEN YMIN=Y;
     ELSE GOTO M5;
          L:END K;
     IF H>0.01 THEN DO;
     M5:IF H>0.01 THEN GOTO L;END;
     X0=X-0.2;H=0.01;GOTO L;END;
     PUT DATA(YMIN,YMAX);END W150;
```

3.232

```
W151:PROC OPTIONS(MAIN);
     DCL Z(20) FLOAT;
     GET LIST(A,B,OMEGA,FI);
     X=0;K=1;
     T3:YMAX=-1E19;YMIN=1E19;
     A5:Y=A*EXP(-B*X)*SIN(OMEGA*X+FI);
     IF Y>YMAX THEN YMAX=Y;ELSE
     IF Y<YMIN THEN YMIN=Y;
     ELSE GOTO R1;
     X=X+0.1;GOTO A5;
   R1:Z(K)=YMAX;Z(K+1)=YMIN;
     K=K+2;
       IF X<=5.05 THEN GOTO T3;
     PUT LIST((Z(I)DO I=1 TO K-1));
     END W151;
```

3.238

```
W196:PROC OPTIONS(MAIN);
     DCL(X,Y ,Z )FIXED(6,2),
     S FIXED(8,2);
     DCL FUN ENTRY (FLOAT,FLOAT)
     RETURNS(FIXED(8,2));
     GET LIST(X,Y,Z );
     S=FUN(X,Y )+FUN(X,Z)+FUN(Y,Z);
     PUT DATA(S);
     END W196;
FUN: PROC(A,B) RETURNS(FIXED(8,2));
     D=SQRT(A*A+B*B+SIN(A*B)**2);
     RETURN(D);
     END FUN;
```

3.239

```
W200:PROC OPTIONS(MAIN);
     DCL C FIXED(8),
     FACT ENTRY  RETURNS(FLOAT(6));
     GET LIST(N,M);
     C=FACT(N)/FACT(M)/FACT(N-M);
     PUT DATA (C);END W200;
FACT:PROC(K) RETURNS(FLOAT(6));
     IF K<0 THEN RETURN(0);
     IFK=0 THEN RETURN(1);
     R1=1; DO I=1 TO K;
     R1=R1*I;END;
     RETURN(R1);END FACT;
```

3.240

```
W201:PROC OPTIONS(MAIN);
     DCL Z(80)FIXED(8,3),
     X(60)FIXED(8,3)DEF(Z),
     Y (75)FIXED(8,3)DEF(Z),
     SRED ENTRY  RETURNS(FIXED(8,3));
     GET EDIT(X)(SKIP,10F(8,3));
     S=SRED(X,60);
     PUT DATA(S)SKIP(2);
     GET EDIT(Y )(SKIP,10F(8,3));
     S=SRED(Y ,75);
     PUT DATA(S) SKIP(2);
     GET EDIT(Z)(SKIP,10F(8,3));
     S=SRED(Z,80);
     PUT DATA(S)SKIP(2);
     END W201;
SRED:PROC(A,N) RETURNS(FIXED(8,3));
     DCL A(*) FIXED(8,3),N FIXED(2);
     M=0;S=0;
     L:DO I=1 TO N;
     IF A(I)>0 THEN DO;
     S=S+A(I);M=M+1;END;
     END L;
     RETURN(S/M);
     END SRED;
```

3.241

```
W260:PROC OPTIONS(MAIN);
     DCL A(10,20)FIXED(8,2)EXT,
     SG ENTRY;
     GET EDIT(A)(SKIP,5F(8,2));
     DO J=1 TO 20;
     S=SG(J);
     PUT EDIT(S)(SKIP,E(12,4));END;
     END W260;
SG:PROC(J);
     DCL A(10,20) FIXED(8,2)EXT;
     Z=1;N=0;
CIKL:DO I=1 TO 10;
     IF A(I,J)>0  THEN DO;
     Z=Z*A(I,J);N= N+1;END;
     END CIKL;
     RETURN(Z**(1/N));
     END SG;
```

3.242

```
W204:PROC OPTIONS(MAIN);
     DCL(A(40),B(60))FLOAT(6),
     (SUM,AMIN)ENTRY ,
     (SQRT,ABS) BUILTIN;
     GET LIST(A,B);
     Z =(SUM(A,40,SQRT,AMIN)+
     SQRT(SUM(B,60,ABS,AMIN)))/2;
     PUT DATA(Z);
     END W204;
SUM:PROC(X,N,F,F1);
     DCL(F,F1)ENTRY ,X(*)FLOAT(6),(I,
     N)DEC FIXED(2);
     S=0; DO I=1 TO N;
     S=S+F(X(I))/F1(X,I);END;
     RETURN(S);
     END SUM;
AMIN:PROC(X,N);
     DCL N DEC FIXED(2),
     X(*)FLOAT(6);
     XMIN=X(1);
     DO I=1 TO N;
     IF X(I)<XMIN THEN
     XMIN=X(I);END;
     RETURN(XMIN);END AMIN;
```

3.243

```
W205:PROC OPTIONS(MAIN);
     DCL A(30,30)FIXED(4,1),
     (M,N,KOL) FIXED(3,0);
     GET LIST(M,N,((A(I,J)DO I=1 TO N)
     DO J=1 TO M));
     CALL F(A,N,M,KOL,S);
     PUT EDIT(S,KOL)(E(12,4),F(9));
     END W205;
F:PROC(X,K,L,IK,S);
     DCL X(*,*) FIXED(4,1),
     (K,L,IK)FIXED(3);
     S=0;IK=0;
  M1:DO I=1 TO K;
  M2:DO J=1 TO L;
     IF X(I,J)>0 THEN DO;
     S=S+X(I,J);IK=IK+1;END;
     END M2;END M1;
     RETURN;END F;
```

3.244

```
W206:PROC OPTIONS(MAIN);
     DCL(A,B,C)FIXED(5,2);
     DCL SQR ENTRY((5)FLOAT(6));
     GET DATA(A,B,C);
     CALL SQR(A,B,-1.5,X1,X2);
     CALL SQR(2.,-1.,C,Y1,Y2);
     U=(EXP(X1+Y1)-EXP(X2+Y2));

     PUT DATA(U);
     END W206;
SQR:PROC(A,B,C,Z1,Z2);
     D=B*B-4*A*C;
     IF D<0 THEN DO;
     Z1=0;Z2=0;RETURN;END;
     Z1=(-B+SQRT(D))/(2*A);
     Z2=(-B-SQRT(D))/(2*A);
     RETURN;
     END SQR;
```

3.245

```
W207:PROC OPTIONS(MAIN);
     DCL (A(30,30),B(30))FIXED(8,2),
     (N,M)DEC FIXED(2),
     SUM ENTRY(FLOAT(6),
     (2)BIN(15),FLOAT(6));
     GET LIST(N,M,((A(I,J) DO I=1 TO N)
     DO J=1 TOM));
     CALL SUM(A,N,M,B);
     PUT EDIT(B(I) DO I=1 TON))
     (SKIP,10F(12,2));
     END W207;
SUM:PROC(X,N1,M1,Y);
     DCL (X(*,*),Y(*)) FLOAT(6);
     L:DO I=1 TO N1;S=0;
     U:DO J=1 TO M1;S=S+X(I,J);
     END U;Y(I)=S;END L;
     RETURN;END SUM;
```

3.246

```
W208:PROC OPTIONS(MAIN);
     DCL(X,Y,A1,A2,A3,R1,R2) FIXED(6),
     SRAV ENTRY((4) FIXED(10));
     GET DATA(X,Y);
     CALL SRAV(X,Y,A1,A2);
     M8:CALL SRAV(A1-A2,A2,R1,R2);
     A3=R1-R2;

     IF A2-A3 ¬=0 THEN DO;
     A1=A2;A2=A3;GOTO M8;END;
     PUT DATA(A3);
     END W208;
SRAV:PROC(X,Y,AMAX,AMIN);
     DCL(X,Y,AMAX,AMIN) FIXED(10);
     AMAX=X;AMIN=X;
     IF Y>AMAX THEN AMAX=Y;
     ELSE AMIN=Y;
     RETURN;END SRAV;
```

3.247

```
W209:PROC OPTIONS(MAIN);
     DCL(A(21),B(40),C(41)),
     FACT ENTRY RETURNS(FIXED(15)),
     SUM ENTRY(,(3)BIN FIXED,);
     GET LIST(A,B,C);
     CALL SUM(A,   3,2,21,X1);
     CALL SUM(B,   2,2,40,X2);
     CALL SUM(C,   2,1,41,X3);
     Z=X1+X2+X3;
     PUT DATA(Z);
     END W209;
SUM:PROC(X,  NO,NH,NM,Z);
     DECLARE(X(*),Z)FLOAT(6),
     FACT ENTRY RETURNS(FIXED(15));
     Z=1;
     DO I=NO TO NM BY NH;
     Z=Z*X(I);END;
     Z=Z/FACT(NM);
     RETURN;END SUM;
FACT:PROC(N) RETURNS(FIXED(15));
     K=1;DO I=1 TO N;
     K=K*I;END;
     RETURN(K);
     END FACT;
```

3.248

```
W210:PROC OPTIONS(MAIN);
     DCL T(200) FIXED(8,3) EXT,
     T1(200) FIXED(8,3) DEF(T),
     TMIN FIXED(8,3),
     (MX,DX) FIXED(10,3) EXT,
     N FIXED(3,0) EXT;
     GET LIST(T);
     N=200; CALL STATIC;
     PUT EDIT(MX,DX)(SKIP,2F(15,3));
  M8:DO K=1 TO 199;
     TMIN=T(K);IMIN=K;
     L:DO I=K+1 TO 200;
     IF T(I)<TMIN THEN DO;
     TMIN=T(I);IMIN=I;END;
     END L;T(IMIN)=T(K);
     T(K)=TMIN; END M8;
     R1:DO I=1 TO 199;
     T1(I)=T(I+1)-T(I);END R1;
     N=199;CALL STATIC;
     PUT DATA(MX,DX)SKIP(2);
     END W210;
STATIC:PROC;
     DCL T(200)FIXED(8,3)EXT,
     (MX,DX)FIXED(10,3)EXT,
     N FIXED(3,0)EXT;
     S=0; DO I=1 TO N;
     S=S+T(I);END;
     S=S/N; D=0;
     DO I=1 TO N;
     D=D+(T(I)-S)**2;END;
     DX=D/N; MX=S;RETURN;
     END STATIC;
```

ОГЛАВЛЕНИЕ

Владимир Евтихиевич Алексеев, Анатолий Сергеевич Ваулин, Галина Борисовна Петрова

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
В ИНЖЕНЕРНЫХ И ЭКОНОМИЧЕСКИХ РАСЧЕТАХ
СБОРНИК ЗАДАЧ И УПРАЖНЕНИЙ

Заведующая редакцией Н. И. Хрусталева. Редактор Л. П. Андрианова. Младший редактор Т. Ф. Артюхина. Художник А. А. Акимов. Художественный редактор В. И. Мешалкин. Технический редактор Р. С. Родичева. Корректор Г. И. Кострикова

ИБ № 4204

Изд. № СТД-388. Сдано в набор 06.02.84. Подп. в печать 05.06.84. Т-05448. Формат 60×90 1/16. Бум. тип. № 2. Гарнитура литературная. Печать высокая. Объем 8,5 усл. печ. л. 8.75 усл. кр.-отт. 10,48 уч.-изд. л. Тираж 80 000 экз. Заказ № 228. Цена 35 коп.

Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14

Московская типография № 8 Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли, 101898, Москва, Центр, Хохловский пер., 7.