Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Строительная механика»

Б.П. Солодов

СБОРНИК ЗАДАЧ С РЕШЕНИЯМИ ПО ПРОГРАММИРОВАНИЮ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ Fortran PowerStation

Методическое пособие

Минск
БНТУ
2011

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Строительная механика»

Б.П. Солодов

СБОРНИК ЗАДАЧ С РЕШЕНИЯМИ ПО ПРОГРАММИРОВАНИЮ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ Fortran PowerStation

Методическое пособие по дисциплине «Информатика»
для студентов вузов специальности
1-70 02 01 «Промышленное и гражданское строительство»
заочной формы обучения

Минск
БНТУ
2011

УДК 004.42+004.438 (076,2)
ББК 32.81я7
С 60

Р е ц е н з е н т ы:
А.А. Борисевич, В.В. Павловец

Солодов, Б.П.

С 60 Сборник задач с решениями по программированию на алгоритмическом языке Fortran PowerStation: методическое пособие по дисциплине «Информатика» для студентов вузов специальности 1-70 02 01 «Промышленное и гражданское строительство» заочной формы обучения / Б.П. Солодов. – Минск: БНТУ, 2011. – 125 с.

ISBN 978-985-525-331-1.

Методическое пособие представляет собой курс практических занятий по программированию на алгоритмическом языке Fortran PowerStation в соответствии с программой курса информатики. Учебный материал излагается в виде последовательности задач; описание операторов языка дается по мере необходимости. Используется минимально необходимое число операторов языка, позволяющее программировать решение задач математики, геометрии, механики. Пособие может быть использовано студентами вузов других технических специальностей.

УДК 004.42+004.438 (076.2)
ББК 32.81я7

ISBN 978-985-525-331-1

Предисловие

Данное пособие предназначено для студентов специальности «Промышленное и гражданское строительство» заочной формы обучения, но может оказаться полезным и студентам дневной формы обучения.

Учебным планом предусмотрено изучение дисциплин «Теоретическая механика», «Сопротивление материалов», «Численные методы», «Строительная механика», «Деревянные и металлические конструкции», «Железобетонные конструкции». Освоение этих предметов требует хорошей математической подготовки и освоения курса «Информатика» на таком уровне, который позволит применять полученные знания для расчета домашних заданий по этим учебным дисциплинам.

Курс изложен по принципу «от простого к сложному», от задачи – к задаче. Операторы языка Фортран приводятся последовательно в соответствии с темами. Используется минимально необходимое число операторов языка, обеспечивающее возможность программирования любых инженерных задач.

В первом разделе дано понятие «Информатика», список учебной литературы с кратким описанием каждой книги, приведены ***краткие сведения об архитектуре*** персонального компьютера, электронной памяти, ***о машинных командах.*** При ***первом*** изучении предмета ***этот материал, если он окажется сложным для понимания, можно пропустить*** и вернуться к нему позднее. В каждой учебной группе находится несколько любопытных студентов, которым просто интересно узнать, как «думает» компьютер. А некоторые студенты вообще не могут нормально воспринимать понятие «программа» без достаточно четкого представления о машинной программе. К сожалению, в большинстве современных учебников этот вопрос освещается очень скудно. Далее приводятся сведения о простых и индексированных переменных, операторах описания. Очень подробно излагаются все варианты вывода текста, приводится таблица системных функций. Подробно на конкретных примерах изложено понятие ***алгоритма***, различные виды алгоритмов и техника их реализации на алгоритмическом языке Fortran PowerStation. Подробно дана техника реализации и отладки

Fortran-программы на конкретном простейшем примере. Приведены примеры программ ***задач следующих типов:*** вычисления по формулам, задачи геометрии, численное исследование нелинейных функций, операции над элементами массивов чисел, ряд задач из курса линейной алгебры, который изучается на первом курсе, вычисление параметров геометрических фигур регулярной структуры, а также простейшие задачи механики из курса «Теоретическая механика», который изучается студентами параллельно с курсом «Информатика».

Данное пособие может быть использовано и студентами вузов *других технических специальностей*, т.к. курсы математики в них имеют много общего с курсом математики для инженеров-строителей.

1. Общие сведения

1.1. Понятие об информации и информатике

Слово ***«информация»*** в переводе с латинского означает разъяснение, изложение сведений о чем-либо. Под это определение попадает все, что мы слышим, видим, читаем, осязаем. **Все сведения** о том, что мы видим, слышим и осязаем **можно** с помощью технических средств **зафиксировать** на различного рода носителях информации, занести в компьютер в виде ***файлов*** и выполнить **обработку** полученной информации с помощью специальных программ. ***Файл*** – это определенная информация, представленная в форме, указывающей компьютеру, что с ней делать.

Например, ***программный файл*** содержит набор команд, каждая из которых выполняет какую-либо операцию над данными, расположенными в других файлах. Каждый файл имеет ***имя,*** состоящее **из двух слов**, разделенных точкой. Первое слово является собственно именем, которое придумывает автор программы, а второе слово – ***тип файла***. Типы файлов имеют типовые имена, которые хранятся в памяти компьютера. Например, файл с именем ROSS.f90 компьютер воспримет как текст программы на алгоритмическом языке Фортран и что его можно транслировать, т.е. переводить на язык компьютера. А вот файл

ROSS.exe – это программный файл, составленный в машинных кодах, готовый к реализации. О машинных кодах будет сказано ниже.

Информатика *– это наука о законах и методах измерения, хранения, переработки и передачи информации с использованием компьютера, который является средством, позволяющим реализовать новые информационные технологии, качественно отличающиеся от прежних уровнем автоматизации и интеллектуализации информационных процессов.*

Современная информатика развивается **по трем направлениям**:

1. Разработка методов и алгоритмов ***автоматизированного сбора***, ***хранения и переработки информации.***
2. Разработка ***методов и алгоритмов обработки и преобразования информации.***
3. Разработка ***технологии и вычислительной техники***, позволяющих развивать первые два направления.

Настоящий курс относится ко второму направлению и включает языки программирования и разработку программ решения инженерных задач.

Алгоритмическому **языку Фортран более** 50 **лет**. Но такие ведущие компании, как IBM, Microsoft, Intel считают своим долгом иметь в своем арсенале новейшие компиляторы Фортрана. Этот язык очень удобен для реализации вычислительных алгоритмов. Для суперкомпьютеров компиляторы Фортрана разрабатываются в числе первых. На Фортране написано огромное число программ для решения научно-технических задач.

1.2. Учебная литература

1. Рыжиков, Ю. Программирование на Фортране PowerStation **для инженеров: практическое руководство / Ю. Рыжков. – СПб.: КОРОНА принт, 1999. – 160 с.**

Подробно описан язык, работа с простыми переменными и массивами, а также ряд вычислительных методов, часть из которых изучается студентами специальности ПГС. Книга является хорошей, полной инструкцией по программированию, но **не является учебником**.

2. Белецки, Я. Фортран-77 / Я. Белецки. – М.: Высшая школа, 1991. – **207 с.**

В этой небольшой книге сжато, но полно изложена версия Фортрана 1977 года. Большинство операторов можно использовать в Фортране PowerStation. Недостаток – отсутствие примеров программирования задач математики.

3. Павловец, В.В. Информатика. Программирование на Фортране / В.В. Павловец. – Минск: Асконто, 2006. – 2005 с.

Подробно на многочисленных примерах изложены практически все приемы программирования на языке Фортран-90 и Фортран PowerStation. Почти все, что там изложено, можно использовать студентам-строителям. Но практические примеры относятся к области расчета электрических цепей. Недостаточно примеров решения математических задач. Особо важным положительным отличием этой книги является наличие обширных таблиц сообщений транслятора об ошибках на русском языке на 80 страницах. Книга является хорошим учебным пособием и может использоваться студентами специальности «Промышленное и гражданское строительство» и инженерами.

4. **Трепачко**, **В.М. Краткий курс программирования на алгоритмическом языке** Fortran PowerStation: **методическое пособие по дисциплине «Информатика» для студентов специальности**
1-70 02 01 «Промышленное и гражданское строительство» / В.М. Трепачко. – Минск: БНТУ, 2006. – 110 с.

Достаточно подробно изложены вопросы:

– этапы реализации инженерных задач на компьютере;

– структура программы;

– все операторы языка Фортран, которые, могут пригодиться студенту и будущему инженеру, с примерами их применения;

– примеры выполнения контрольных заданий с блок-схемами: численные исследования функций, операции над массивами, решение уравнений, решение задач модульной структуры.

Это первое учебное пособие, которое предназначено для студентов специальности «Промышленное и гражданское строительство».

1.3. Понятие об устройстве персонального компьютера и принцип его работы

Этот материал не окажет существенного влияния на изучение программирования, но он поможет, если вам трудно будет понимать смысл операторов Фортрана.

Архитектура компьютера – логическая организация, структура и ресурсы компьютера, которые может *использовать программист*. *Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера.*

Основные устройства компъютера – процессор, оперативная электронная память, внешняя электронная память, устройство ввода информации, устройство вывода информации.

Процессор – интегральная микросхема, изготовленная на основе небольшого (иногда меньше 1 $см^2$) куска кремния. Кремний – один из самых распространенных элементов на земле. В обычных условиях кремний не проводит ток. Но если в него добавить небольшое количество примесей (например, бор или фосфор), то свойства его кристаллической структуры меняются.

Микросхема состоит из многих отдельных слоев, в которых есть резисторы (сопротивления), конденсаторы, транзисторы и другие элементы.

В процессор входят **арифметическое и логическое устройства**, устройство управления, регистры, сумматор, в которых хранятся данные, поступающие в процессор, и выходящие из него. Важнейшей характеристикой процессора является *тактовая частота* – количество операций сложения и вычитания за секунду.

Из-за малых размеров процессор называют **микропроцессором,** в отличие от процессоров вычислительных машин, которые использовались до появления персональных компьютеров.

Арифметическое и логическое устройства выполняют анализ содержимого машинных команд и выполняют различные операции над информацией, хранящейся в ячейках памяти. Понятие о машинных командах и ячейках памяти дано ниже.

Устройство управления управляет всеми узлами компьютера и внешними устройствами. **Регистры** предназначены для временного хранения информации внутри процессора перед выполнением

заданной операции. Например, из ячеек памяти извлекаются два числа и заносятся в регистры, после чего арифметическое устройство выполняет сложение этих чисел и результат пересылается по указанному в машинной команде адресу.

Сумматор предназначен для временного хранения в процессоре результата выполнения той операции, которая выполнена арифметическим или логическим устройством.

Внешняя (долговременная) память – это место длительного хранения данных (программ, результатов расчетов, текстов и т.д.), не используемых в данный момент в оперативной памяти компьютера. Внешняя память, в отличие от оперативной, является энергонезависимой. Носители внешней памяти, кроме того, обеспечивают транспортировку данных в тех случаях, когда компьютеры не объединены в сети (локальные или глобальные). Для работы с внешней памятью необходимо наличие *накопителя* (устройства, обеспечивающего запись и (или) считывание информации) и устройства хранения – *носителя.*

Основные виды накопителей:

– накопители на гибких магнитных дисках (НГМД) – дискеты;
– накопители на жестких магнитных дисках (НЖМД);
– накопители на магнитной ленте (НМЛ);
– накопители CD-ROM, CD-RW, DVD;
– флэш-память.

Оперативная электронная память

Ее удобно представить в виде очень длинного ряда ячеек. Каждая ячейка памяти в свою очередь состоит из устройств для хранения наименьшей единицы памяти – бита информации. ***Один бит*** – это единица или ноль. Если электрический заряд имеется, то содержимое бита равно 1, если заряд отсутствует, то это ноль. Биты объединены в ***байты***, каждый из которых содержит 8 бит.

Одна ячейка памяти (слово) может содержать 2, 4, 8 и более байт. ***Объем памяти*** носителя информации принято подсчитывать в байтах. **1 байт** = 8 бит. **1 килобайт** = 1024 байта, **1 мегабайт** содержит 1024 килобайта, **1 гигабайт** содержит 1024 мегабайта. Для массового потребления уже используются носители

информации, объем которых измеряется **терабайтами**, 1 терабайт = 1024 гигабайт. Любой символ кодируется сочетанием единиц и нулей. Например, числа кодируются с помощью *двоичного* кода (каждый бит имеет два состояния: 1 или ноль). Поэтому «с точки зрения» компьютера числа имеют вид 0, 1, 10, 11, 100, 101 и т.д. Арифметика компьютера на первый взгляд имеет забавный вид: 0 + 1 = 1, 1+ 1 = 10, 10 + 1 = 11, 11 + 1 = 100. Такая арифметика носит название *двоичной системы счисления*. Привычная нам арифметика, использующая цифры 1, 2, … 9, 10 в терминах программирования носит название *десятичной системы счисления*.

Здесь для сведения мы приведем таблицу ***трех систем счисления*** (табл. 1.1).

Таблица 1.1

Двоичная	Восьмеричная	Десятичная
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	10	8
1 001	11	9
1 001	12	10

Из этой таблицы видно, что между двоичной и восьмеричной системами счисления очень удобная связь, позволяющая двоичные коды изображать восьмеричными числами. И еще используется шестнад-цатиричная система счисления, в которой десятичное число 16 является шестнадцатиричной десяткой. Но здесь мы остановимся, так как для практики программирования в пределах настоящего курса информатики это существенного значения не имеет. В учебниках по программированию прошлых лет подробно описаны различные системы счисления и способы перевода чисел из одной системы счисления в другую.

Часть ячеек памяти содержит ***машинные команды***, с помощью которых ***микропроцессор*** выполняет различные операции над информацией, хранящейся в других ячейках памяти – данными. Все машинные команды и данные находятся в памяти компьютера в виде ***двоичных кодов. Машинное слово*** *– набор из двух, четырех или восьми последовательных байтов, обрабатываемый аппаратной частью вычислительной системы как единое целое.*

Понятие о машинной команде

Компьютеры разных типов имеют разную структуру команд. Всякая машинная команда состоит из ***кода операции и адресной части***.

Машинная команда представляет собой закодированное по определенным правилам указание микропроцессору на выполнение некоторой операции или действия. Совокупность машинных команд образует ***рабочую машинную программу***. Команды хранятся в ячейках памяти ***в двоичном коде.***

В общем случае ***команда содержит***:

– код выполняемой операции;

– указания по определению ***операндов (их адресов***);

– указания по размещению получаемого результата.

Результат выполнения команды вырабатывается по точно определенным правилам, заложенным в конструкцию компьютера. В зависимости от количества операндов различают ***одноадресные, двухадресные***, ***трехадресные*** и переменноадресные команды.

Код операции обычно представляет собой двух- или трехзначное число. Например, 10 – сложение; 20 – вычитание; 30 – умножение и т.д. Следом за кодом операции расположен *номер индексной ячейки*, о которой будет сказано ниже. Далее расположены ***адреса ячеек памяти***. Если указан адрес только одной ячейки памяти, то это ***одноадресная машина***, если указаны адреса двух ячеек памяти, то это ***двухадресная*** машина, а если указаны адреса трех ячеек памяти, то это ***трехадресная*** машина. Работа компьютера расписана в ***рабочей машинной программе*** по действиям. ***Пусть, например****, необходимо подготовить рабочую машинную программу для вычислений по формуле* $R = (A + B) \cdot C$. Буквы R, A, B, C являются

символическими именами адресов ячеек памяти, в которых предстоит хранить соответствующие числа.

Перед составлением рабочей машинной программы необходимо *назначить номера ячеек* памяти с именами *A, B, C, R*; пусть это будут соответственно ячейки с номерами 101, 102, 103, 104. Затем следует *назначить номер первой ячейки* памяти для машинной программы. Предположим, что числа в ячейки памяти с именами *A, B, C, R* уже занесены или сформированы заранее с помощью предыдущих команд, и номер (он же и адрес) последней команды 1276 в восьмеричной системе счисления. Тогда ***программа*** для процессора ***одноадресной машины*** должна содержать следующую последовательность операций:

1277) Из ячейки памяти 101 извлечь число (копию его) и поместить во внутренний регистр № 1.

1300) Из ячейки памяти 102 извлечь число (копию его) и поместить во внутренний регистр № 2.

1301) Выполнить сложение содержимого регистров и оставить результат в регистре, например, № 1.

1302) Из ячейки памяти 103 извлечь число и разместить в регистре № 2.

1303) Выполнить умножение двух чисел, находящихся в регистрах № 1 и № 2, результат оставить в сумматоре.

1304) Результат из сумматора переслать в ячейку памяти 104.

Двухадресная машина может выполнить все эти действия с помощью трех команд:

1277) 10 00 0101 0102 – число из ячейки памяти 101 складывается с числом из ячейки 102, а результат заносится в ячейку 102.

1300) 30 00 0102 0103 – выполняется перемножение чисел, расположенных в ячейках памяти 102 и 103, результат остается по второму адресу, т.е. в ячейке с номером 0103.

1301) 10 00 0103 0104 – число из ячейки 103 пересылается в ячейку 104.

Как видим, неудобство двухадресной машины состоит в том, что второй адрес ячейки памяти портится и приходится записывать больше команд, чтобы этого не происходило.

Трехадресная машина для решения этой же задачи использует две команды

1277) 10 00 0101 0102 0104 – выполнить сложение содержимого ячеек 101 и 102, результат временно переслать в ячейку 104.

1300) 30 00 0104 0103 0104 – выполнить умножение содержимого ячеек 104 и 103, результат переслать в ячейку 104.

Отметим, что приведенные здесь ***коды операций 10 и 30 необязательно*** используются в современных вычислительных машинах. Эти коды соответствуют электронной вычислительной машине «Минск-22», которая была популярна в Советском Союзе в 60-е годы.

Операция умножения выполняется путем многократного сложения, а операция ***деления*** – путем многократного вычитания. Важ-нейшей операцией, которую выполняет микропроцессор, является «***королева» всех команд – логическая операция сравнения*** содержимого двух ячеек памяти. Эта операция позволяет программировать сложнейшие логические рассуждения, которые компьютер может повторять по заданной программистом схеме.

Назначение упомянутых выше ***индексных ячеек памяти*** – обеспечить автоматическое изменение адресов ячеек памяти перед выполнением операции. Вот как выглядит индексная ячейка с номером 01 для двухадресной машины:

01) 00 00 0004 0004

В этом случае команда номер 1277 с использованием индексной ячейки 01 выполнится следующим образом:

10 00 0101 0102 + 00 00 0004 0004 = 10 00 0105 0106

Т.е. сначала изменяются адреса ячеек памяти, затем по коду операции 10 выполняется сложение содержимого ячеек памяти с измененными адресами ячеек 105 и 106.

Изменение содержания индексной ячейки автоматически изменяет адреса ячеек памяти, над которыми производится операция. Это позволяет **обрабатывать огромные массивы ячеек памяти очень малым числом** команд. Следует отметить, что здесь дано лишь примерное описание ячейки памяти **стандартной длины** для машин прежних поколений. Фактически длины ячеек памяти **современных** машин могут быть заданы **разными.** А машинная команда может иметь дополнительный адрес – ***адрес следующей команды.*** Это позволяет при отладке программы сохранять ее не сплошным потоком команд, а фрагментами. Любой файл, в конце концов, оказывается

«размазанным», составленным из лоскутков, расположенных в разных местах памяти. Однако есть программы, которые собирают эти лоскутки и объединяют их в сплошные без пробелов файлы. Объединение совокупности битов в ячейки памяти происходит не конструктивно, а ***логически.*** Но эта тема уже выходит за рамки данного пособия.

Вышеупомянутый ***микропроцессор*** «помнит» все коды операций и умеет передавать часть команд исполнительным устройствам ввода и вывода. На любую операцию предусмотрен свой код. Современные программы построены по иерархическому принципу, позволяющему формировать ***подпрограммы***, которые способны с огромной скоростью выполнять типовые операции. Когда были разработаны вычислительные машины, появились и ***программисты***, которые разрабатывали машинные программы сразу в машинных кодах. А поскольку держать в памяти многочисленные коды операций было утомительно, ***появились первые языки*** – системы символического кодирования. В них коды операций были заменены легкозапоминающимися словами, а номера ячеек памяти заменили ***именами.***

Составленную таким образом программу первые ***программы-трансляторы*** переводили в машинные программы. Но программист при этом сам распределял машинную память, точно определяя, где надо разместить группы данных и место для машинной программы.

Потом появились ***алгоритмические языки***, максимально приближенные к человеческому, и соответствующие программы-транс-ляторы. Вычислительные машины имели внушительные размеры. Одна из первых вычислительных машин «Проминь», рассчитанная на 100 ячеек памяти для машинных команд и 100 ячеек памяти для числовых данных, была размером с письменный стол. Вычислительная машина «Минск-22» имела оперативную память в 10 000 ячеек памяти и 12 лентопротяжных механизмов для записи на магнитные ленты, ширина которых была примерно 6 см. Она занимала комнату примерно 6×8 метров и комнату для обслуживающих ее инженеров. Впоследствии появилась ЭВМ серии ЕС (электронная вычислительная машина единой системы), длина которой была около 2 метров. Эта машина была способна обслужить несколько классов, в каждом из которых находилось до 10 дисплеев,

связанных с вычислительной машиной многожильными проводами. Программы уже писали на достаточно совершенных алгоритмических языках.

Одним из самых популярных был Фортран, отличающийся простотой синтаксиса и высокой скоростью трансляции. Появление персональных компьютеров произвело революционные изменения во всей практике программирования.

Алгоритмический язык – формализованный язык для однозначной записи алгоритмов. Состоит из набора символов (алфавит алгоритмического языка), синтаксических правил и семантических определений. Алгоритмический язык (как и любой другой язык) образуют три его составляющие: алфавит, синтаксис и семантика. ***Алфавит*** – это фиксированный для данного языка набор основных символов, т.е. «букв алфавита», из которых должен состоять любой текст на этом языке – никакие другие символы в тексте не допускаются. ***Синтаксис*** – это правила построения фраз, позволяющие определить, правильно или неправильно написана та или иная фраза. Точнее говоря, синтаксис языка представляет собой набор правил, устанавливающих, какие комбинации символов являются осмысленными предложениями на этом языке. ***Семантика*** определяет смысловое значение предложений языка. Являясь системой правил истолкования отдельных языковых конструкций, семантика устанавливает, какие последовательности действий описываются теми или иными фразами языка и, в конечном итоге, какой алгоритм определен данным текстом на алгоритмическом языке.

Логически завершенную часть информации объединяют в **файл,** которому программист назначает **имя**. Вслед за именем файла через точку следует **расширение имени**, которое относит файл к тому или иному стандартному типу. Например, Вы подготовили ***текст программы*** на алгоритмическом языке Fortran PowerStation по имени Ruta. Полное его имя с расширением будет иметь вид Ruta.f90. Затем Вы запускаете эту программу на трансляцию. После первого этапа трансляции, если транслятор не обнаружит ошибок, будет сформирован файл Ruta.obj – ***объектный модуль***, в котором Ваша программа имеет вид полуфабриката – совокупность малых модулей, каждый из которых записан с нулевого адреса. После

следующего этапа трансляции будет сформирована ***машинная программа*** – файл Ruta.exe, которая уже адаптирована к работе с того адреса, который ей определил транслятор. Если Вы набираете текст в редакторе Word, то формируется файл с расширением *.doc (document). Каждый файл оформляется стандартным образом и может быть записан на внешний носитель. По расширению имени компьютер «узнает» что это за файл и что с ним делать.

2. Представление чисел в компьютере

В своих вычислениях мы пользуемся десятичной системой счисления, а компьютер – двоичной системой счисления. Поэтому с помощью специальных подпрограмм десятичные числа, вводимые с клавиатуры или из другого носителя информации, переводятся в двоичные аналоги. При выводе на экран двоичные коды чисел переводятся в десятичные числа. **Целыми числами** обычно обозначаются объекты, которые не могут быть дробными в принципе. Например, количество студентов 25, число столбцов в матрице 10, номер элемента массива 17 не могут быть дробными. В памяти компьютера такие числа располагаются так:

0000025, 0000010, 0000017.

Слева от ненулевых цифр содержимое ячейки памяти дополняется нулями. Большие целые числа могут изображаться с использованием показателя степени, например, 1E+7. В стандартной ячейке памяти наибольшее целое число может достигать по модулю примерно $2 \cdot 10^9$. Целые числа относятся к **типу** Integer (в переводе с английского языка «целое число»). Будем помнить, что в ячейках памяти хранятся не десятичные числа, а их двоичные аналоги.

Вещественными числами (**тип** Real) измеряются размеры объектов, вес, объем и все, что может выражаться дробными числами. Любое вещественное число, **даже если оно записано как целое**, в памяти компьютера преобразуются в форму с плавающей десятичной точкой. Такие числа *лучше всего всегда писать с десятичной точкой*, чтобы исключить возможные ошибки.

Число 56.0 приводится к виду $0.5600000 \cdot 10^2$ и представляется в ячейке памяти в форме с плавающей десятичной точкой и имеет

примерно такой вид: +5600000+02. Естественно, что числа в памяти компьютера представлены в двоичном коде и состоят из единиц и нулей. Первая часть числа +5600000 называется **мантиссой** и содержит в переводе в десятичную систему счисления 7 ***значащих цифр***. Вторая часть числа – +02 – ***показатель степени***.

При выводе на экран или в файл вещественные числа записываются в форме с ***фиксированной*** десятичной точкой или в форме с ***плавающей*** десятичной точкой. Если число не является очень большим или очень малым, то записывать его удобнее в форме с фиксированной десятичной точкой, например: –256.337, –0.00328. *Главные характеристики этих чисел – длина (общее число символов) и число знаков после десятичной точки.* По этим признакам два эти числа для вывода на внешние устройства могут быть описаны так: F8.3, F8.5. Буква F (код) показывает, что число выводится по коду с фиксированной десятичной точкой, затем первая цифра указывает общую длину числа, а вторая цифра после точки указывает на число знаков после десятичной точки. Если Вы не можете заранее предсказать величины чисел, которые предстоит вывести, то лучше выводить их в форме с **плавающей десятичной точкой**. Предыдущие два числа будут преобразованы в следующем виде: $0.2563370 \cdot 10^3$, $-0.3280000 \cdot 10^{-2}$. Они будут выведены на экран или в файл (как Вы укажете в операторе вывода PRINT или WRITE) в следующей форме: –0.256337E+03, –0.328E–02. Как видно из этих примеров, число, выводимое в форме с плавающей десятичной точкой, имеет следующие параметры: *общее число символов, включая возможный знак «минус», и число знаков после десятичной точки.* Поэтому описание этих чисел имеет вид E13.6, E10.3. Здесь буква E является кодом и показывает, что число будет выведено в форме с плавающей десятичной точкой.

3. Простые и индексированные переменные

Как было сказано в первом разделе, компьютер реализует замысел программиста, последовательно выполняя команды машинной программы. Каждая машинная команда содержит адреса ячеек памяти, над содержимым которых по командам выполняются операции. **Адресом ячейки памяти является ее номер.**

В современных условиях программирование вычислительных (и не только) задач выполняется **на алгоритмических языках**, которые вместо адресов (номеров) ячеек памяти используют **имена:** буквы или сочетания букв и цифр как в алгебре. Например, выражение $C = A + B$ на языке программирования является **не уравнением, а командой**: ***содержимое ячейки памяти с именем A сложить с содержимым ячейки памяти по имени B и результат занести в ячейку памяти с именем C***. Поэтому буквы *A, B, C* являются ***именами ячеек памяти***, которые в машинной программе превратятся в ***номера ячеек памяти*** (адреса). Следом за этой командой может быть записана следующая: $C = A \cdot B$. Транслятор ошибку не обнаружит. И только программист должен знать: если он не организовал вывод на экран результата вычислений по первому выражению, то этот результат заменится результатом вычислений по второй формуле, а результат вычислений по первой формуле будет потерян.

Поэтому программа на языке программирования для этих двух выражений должна включать операции:

1. $C = A + B$; вывод на экран значения *C*.
2. $C = A \cdot B$; вывод на экран значения *C*.

Если программист не организует вывод результатов вычислений, то сам компьютер этого не сделает. Он выполнит вычисления и остановится или продолжит выполнять следующие команды, если они имеются. Но решение рассмотренной задачи можно организовать иначе:

1. $C = A + B$.
2. $D = A \cdot B$.
3. Вывод на экран значений *C, D*.

Так как содержимое ячеек памяти в процессе реализации программы может изменяться, то эти буквы называются ***простыми переменными***.

Кроме простых переменных в программировании используются ***индексированные переменные***, которые являются элементами **массивов чисел – одномерных и двумерных**. Примеры таких массивов приведены ниже. Для наглядности у одномерного массива обозначены номера строк, а у двумерного – номера строк и столбцов.

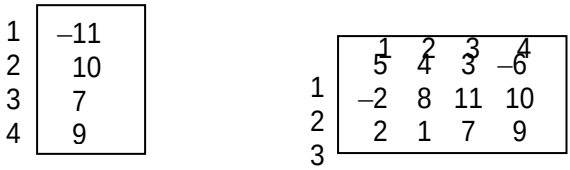


Рис. 3.1. Примеры массивов: одномерного и двумерного

Как одномерные, так и двумерные массивы имеют по одному имени. Назовем приведенные выше массивы именами *V* и *W*, тогда *V(3) = 7* – третий элемент массива *V, W(2, 3) =11* – элемент, расположенный на пересечении 2-й строки и 3-го столбца. Первая цифра в скобках – номер строки, вторая цифра – номер столбца. Эти цифры называются ***индексами*** массивов.

Индексы чаще являются простыми переменными, в этом случае *V(k)* – элемент № *k* одномерного массива *V*, а *W(i, j)* – элемент двумерного массива *W*, расположенный в строке *i*, в *j*-м столбце. Понятно, что для того, чтобы найти эти элементы, *необходимо предварительно вычислить* или задать числовые *значения индексов k, i, j*. А поскольку числовые значения простых переменных **можно изменять** по какому-либо правилу, то имеется возможность использовать в вычислениях все элементы массива ограниченным числом машинных команд.

Транслятор алгоритмического языка Fortan PowerStation располагает **элементы двумерных массивов в памяти компьютера по столбцам.**

В качестве примера предположим, что два приведенных выше массива расположены в памяти компьютера с адресом 3458 (в подсчетах используем десятичные номера ячеек памяти, хотя компьютер оперирует двоичными числами). Тогда участок машинной памяти с этими массивами можно условно представить так, как показано на рисунке 3.2.

Рис. 3.2. Схематичное изображение расположения чисел в памяти компьютера: в одномерном и двумерном массивах

Порядковый номер ***элемента*** *V(k)* ***внутри одномерного массива*** *равен численному значению его индекса, это очевидно. А расположение этого элемента в памяти компьютера равно* ***адресу ячейки памяти перед массивом*** *V* ***плюс*** *k: 3457 + 3 = 3460, что мы и наблюдаем на схеме для элемента V(3).*

Вернемся к нашему ***двумерному массиву*** *W. Порядковый номер элемента W(2,3) в памяти компьютера равен числу столбцов, предшествующих третьему столбцу, умноженному на число элементов столбца, плюс номер строки: Nom = (2*3) + 2 = 8. В общем случае для двумерного массива W, содержащего M строк и N столбцов, порядковый номер элемента W(i,j) равен*

*Nom = (j –1)*M + 1 = (3 – 1)*3 + 2 = 8.*

Тогда расположение элемента W(i, j) в памяти компьютера определяется так: адрес ячейки, предшествующей адресу первого элемента массива, плюс его порядковый номер в массиве, т.е. адрес

элемента W(2, 3) в памяти компьютера равен 3461 + Nom = 3461 + 8 = 3469.

Вот таким образом компьютер находит в своей памяти адреса элементов массивов.

Числовые значения простых переменных и массивов могут быть как ***целыми****, так и* ***вещественными***. Если не указывать их тип, то простые переменные и массивы, имена которых начинаются с одной из букв *I, J, K, L, M, N,* являются простыми переменными и массивами ***целого типа*** *INTEGER*. ***Остальные*** переменные и массивы являются ***вещественными*** – типа *REAL*.

Имя простой переменной или массива ***начинается с буквы***, состоит из одной или нескольких букв латинского алфавита и может включать цифры. Например: *A, A1, NOMER, Index, sila, GG(i), Massa(i,j), a11, a12.* Желательно, чтобы имена не были слишком длинными. Буквы могут быть ***строчными*** или ***прописными***, транслятор их не различает.

4. Операторы описания простых переменных и массивов

Как было отмечено выше, в вычислениях используются целые и вещественные числа. Соответственно, простые переменные и массивы бывают как ***целого, так и вещественного типа***. В большинстве случаев перед их использованием простые переменные можно не описывать, пользуясь типом первой буквы имени, хотя при более строгом отношении к делу, лучше описывать все простые переменные. ***Массивы же описывать необходимо всегда***, так как в результате трансляции машинная программа должна иметь фиксированную длину. Это значит, что при разработке алгоритма вы должны заранее определить наибольшие размеры массивов для своих примеров. Если вы назначите ***размеры массивов*** намного большими, чем вам надо, то память компьютера будет занята множеством пустых ячеек. Отметим, что Фортран допускает динамическое распределение памяти, т.е. размеры массивов можно менять в зависимости от числовых значений исходных данных, но это выходит за рамки нашего пособия.

Оператор *REAL* позволяет описывать простые переменные и массивы ***вещественного типа.*** Например:

REAL L(10), b(0:10), C(–3:5), N(5,10), E(–5:5,–10:15), I, m

В одномерном массиве *L* индекс изменяется от 1 до 10.

В одномерном массиве *b* индекс изменяется от 0 до 10.

В одномерном массиве *C* индекс изменяется от –3 до 5.

В двумерном массиве *N* номер строки изменяется от 1 до 5, а номер столбца от 1 до 10.

В двумерном массиве *E* номер строки изменяется от –5 до 5, а номер столбца от –10 до 15.

Имена простых переменных и массивов можно записывать как строчными, так и прописными буквами или сочетаниями букв и цифр. Но первым символом имени должна быть буква.

В нашем примере простые переменные *I, m* описаны как вещественные, потому что *«по умолчанию»* они подразумеваются целыми, а программисту оказалось удобным использовать именно эти буквы в качестве имен вещественных простых переменных. Например, в сопротивлении материалов момент инерции поперечного сечения принято обозначать буквой *I*, продольные усилия в стержнях – буквой *N*, длины пролетов сооружений часто обозначают буквой *L*. Оператор *INTEGER* позволяет описывать простые переменные и массивы ***целого типа.*** Например:

INTEGER T(10), S(0:10), M(–3:5), N(5,10), F(–5:5,–10:15), r, s

Смысл числовых обозначений в скобках тот же, что и в операторе *REAL*. Часто при обозначении индексов индексированных переменных бывает недостаточно букв *i, j, k, L, m, n,* которые по умолчанию являются целыми. Тогда приходится писать, например *U(r,s)*. В качестве индексов индексированной переменной в данном примере использованы буквы *r, s*, которые по умолчанию используются в Фортране как вещественные простые переменные. Но *для обозначения индексов нельзя использовать вещественные простые переменные*, поэтому эти индексы должны быть описаны как целые (типа *integer)*.

Оператор *DIMENSION* (в переводе на русский язык – «размерность») сегодня уже ***менее актуален***. Он описывает только размерности массивов, после него при необходимости надо добавить операторы *REAL, INTEGER*, например:

DIMENSION A(5), M(–3:5, –2:8); REAL M

Массив *M* в данном примере предусмотрен как вещественный, а «по умолчанию» он относится к целому типу, поэтому рядом (или в следующей строке) должен быть записан оператор *REAL M*, который закрепляет за именем *M* тип вещественного. Но нельзя было в данном случае записать *REAL M(–3:5, –2:8),* так как размер его уже описан оператором *DIMENSION*.

Оператор *DIMENSION* использовался в более ранних версиях Фортрана, когда операторы *INTEGER, REAL* не могли выполнять функции описания размеров массивов. В Фортране и в других языках предусмотрено использование простых переменных и массивов **символьного** типа, которые в обязательном порядке необходимо описывать оператором *CHARACTER* (в переводе с английского – «символ, литера, буква»). Например:

*Character A*10, B(5)*20*
*A = 'massiv R:'; B(1) = 'Matrica F(m*n)'*

Для простой переменной *A* зарезервировано 10 символов; для каждого элемента одномерного массива *B* зарезервировано 20 символов. Если в тексте программы будет записан оператор *PRINT *,A* то на экран будет выведено: *massiv R.* **Не следует** применять в качестве символьной константы **буквы русского алфавита**, он не используется транслятором Фортрана. Вместо русского текста на экране появится набор непонятных символов. В Фортране предусмотрено использование **других операторов описания,** но их могут использовать уже более опытные программисты в сложных и больших по размеру программах. В рамках начального обучения программированию операторов *Real, Integer, Character* вполне достаточно для решения большинства инженерных задач.

5. Вывод текста на экран

Текст выводится на экран при помощи операторов *PRINT* (печатать), *WRITE* (писать), Например *PRINT *, 'Programma Ruta'*. На экране появится текст, который в операторе PRINT заключен в апострафы.

Набор символов, заключенный в апострафы, рассматривается транслятором как ***символьная или текстовая константа***. Символ «*» после слова *PRINT* значит, что для вывода не используется оператор *FORMAT*. Этот же текст можно вывести при помощи оператора *FORMAT:*

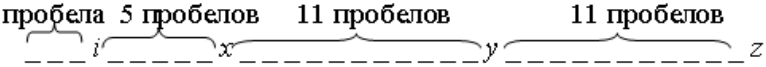
```
PRINT 11
11 FORMAT(1x, 'Programma Ruta')
```

Вывод текста с использованием оператора *FORMAT*

1x – один пробел, является символом управления печатью; означает, что начать печатать следует с новой строки.

Оператор *FORMAT* удобен при выводе текста, содержащего много повторяющихся символов, например, оглавлений таблиц.

Пусть необходимо вывести строку следующего содержания:



Здесь черточка символизирует один пробел, вместо черточек надо выводить пробелы. Вот как выглядит программа вывода этого текста:

```
PRINT 11;     11 FORMAT(1x, 3x, 'i', 5x, 'x', 11x, 'y', 11x, 'z')
```

Эти же задачи можно решить при помощи оператора *WRITE.*

Вот два примера: без использования оператора *FORMAT* и с использованием.

```
WRITE(*,*) 'Programma Ruta'
WRITE(*,11)
   11 FORMAT(1x, 3x, 'i', 5x, 'x', 11x, 'y', 11x, 'z')
```

Первый символ «*» в скобках значит, что текст следует вывести на экран; если вместо первого символа «*» поставить, например, 2, то это будет означать, что текст следует вывести в файл № 2. Имя этого файла программа потребует ввести с клавиатуры. Второй символ «*» имеет тот же смысл, что и в операторе *PRINT* – бесформатный вывод. Если второй символ является цифрой, то это номер метки строки с оператором *FORMAT,* и выводить надо по формату.

Как видно из сказанного, оператор *WRITE* позволяет выводить информацию не только на экран, но и в файл, который впоследствии можно вывести на печать или записать на внешний носитель информации.

Содержание часто встречающихся символьных констант можно заносить в ***простые переменные символьного типа*** (*CHARACTER)*, а затем выводить их на печать.

```
CHARACTER w1*10,w2*32      Простые переменные w1,w2 объявляются
                           символьными длиной 10 и 32 символов
W1='Zadacha 1:';      W2=': Vichislenije znachenij funkcii'
Write(*,*)w1,w2       или  Print *, w1,w2
```

Здесь *Write* – вывести; (*, – на экран; ,*) – без использования оператора *FORMAT; w1,w2* – имена двух символьных констант.

6. Символы языка Fortran PowerStation

Описание символов языка приведено в табл. 6.1.

Таблица 6.1

Символ	Его описание
1	*2*

=	1) Знак присваивания. Выражение $A = 5$ является не равенством, а командой и означает: простой переменной A присвоить число 5. Компьютер число 5 перешлет в ячейку памяти по имени A. Если до этого в ячейке по имени A находилось число, оно исчезнет. Выражение $A = B$ означает: простой переменной A присваивается значение простой переменной B. Компьютер копию содержимого ячейки памяти по имени B перешлет в ячейку памяти по имени A. Прежнее содержимое ячейки памяти с именем A исчезнет
+ – * /	*Знаки сложения, вычитания, умножения и деления*. Пример: $R = (A + B/C)*F - U$

Продолжение табл. 6.1

1	*2*
**	*Знак возведения в степень.* Например, выражение $R = A + B^3$ на Фортране следует записать так: $R = A + B**3$. При $A = 4$, $B = 5$, значение $R = 129.0$. Показатель степени может быть дробным, однако нельзя возводить в дробную степень отрицательное число. $T = (A + B)**(2./3.) = 4.3267487$. В этом случае **десятичные точки** в цифрах показателя степени ставить **необходимо**. Без них результат деления 2/3 преобразуется **в целое**. При этом дробная часть результата отбрасывается без округления, результат становится **равным нулю**. Любое число в степени 0 равно 1. Поэтому при любых значениях A, B значение T **окажется равным единице**. $L = 0**(2/3)$, будет равно целой 1. $w = 0.**(1/2)$ получим 1.0000

.	*Десятичная точка* в десятичных числах. Если вы забудете поставить ее, то во многих случаях число будет воспринято как целое. *Ставьте десятичную точку всегда, когда смысл числа допускает возможность его дробного значения*. Например, число 2, имеющее смысл длины, лучше записать с десятичной точкой. Если вы напишете *C = (3**2 + 4**2)**(1/2),* то результатом будет 1.000000. Поскольку простая переменная *C* по умолчанию является вещественной, то транслятор исправит вашу ошибку и числа 3 и 4 превратит в вещественные. Но 1/2 – результат деления целых чисел – округляется просто: отбрасывается дробная часть, в результате – ноль. А любое число в нулевой степени равно единице. А если запишете *C = (3**2 + +4**2)**(1./2.)*, то результатом будет 5.000000, так как степень 0.5 – это корень квадратный. **Чтобы не заставлять транслятор «теряться в догадках»**, лучше написать *C = (3.**2 + 4.**2)**(1./2.)*, так как он **не всегда замечает ваши ошибки**
.	*Точка в операторе FORMAT* является разделителем в спецификаторах. Отделяет общее число символов выводимого вещественного числа и число знаков после десятичной точки

Окончание табл. 6.1

1	*2*
.	*Точка в логических выражениях* операторов *IF* является разделителем между словами, составляющими логическое выражение
,	*Запятая,* используется в качестве разделителя в операторах описания простых переменных и массивов, в качестве разделителя между индексами в индексированных переменных, является разделителем в операторе *FORMAT*

;	*Точка с запятой* – разделитель между операторами, записанными в одной строке. Разделитель в операторах описания массивов
,	*Апостроф* применяется для обозначения символьных констант например, *F = 'Marija'; Print *,'gromada'*
:	*Двоеточие* используются в операторах описания массивов для задания пределов изменения значений индексов индексированных переменных
!	*Восклицательный знак* используется для записи комментариев в тексте программы. Вся информация справа от этого знака считается комментариями и транслятором не обрабатывается
< >	*Меньше, больше* используются в логических выражениях в операторах *IF*
<= >=	*Меньше или равно, больше или равно* используются в логических выражениях в операторах *IF*. К сожалению, знак равенства в них использовать нельзя, для этого используется выражение *«.EQ.»*
&	*Амперсанд,* используется для переноса на другую строку продолжения длинного выражения. Этот знак следует установить в конце обрезаемой строки и в начале строки продолжения
( )	*Скобки* используются в арифметических выражениях и в операторах

7. Встроенные системные функции, запись математических формул

|x| (x – целое) *iabs(x)*
|x| (x – вещественное) *ABS(x)*
ln x *ALOG(x)*
lg x *ALOG10(x)*
e^x *EXP(x)*
$\sqrt{x}$ *SQRT(x)*
tg x *TAN(x)*
Ctg x *COTAN(x)*
Sin x *sin(x)*
Cos x *COS(x)*
Arcsin x *asin(x)*
Arccos *ACOS(x)*
Arctg x *ATAN(x)*
Устаревшие: *ARSIN(x)*, *ARCOS(x)*

Преобразование целого в вещественное: *FLOAT(x).*

Преобразование вещественного в целое: *ifix(x).*

Числу *a* присвоить знак числа *b: SIGN(a,b).*

Наибольшее из нескольких вещественных чисел: *AMAX1(a,b,c, ...).*

Наименьшее из нескольких вещественных чисел: *AMIN1(a,b,c, ...).*

Наибольшее из нескольких целых чисел: *AMAX0(a,b,c, ...).*

Выделение целой части вещественного числа: *aint(x).*

Для записи тригонометрических выражений в качестве аргументов используются радианы. Для справки:

$\pi = 3.14159;$

$1\rho = 57.296$ *градусов* – один радиан;

1град $= \pi/180$ *радиан* $= 0.0174532;$

$e = 2.17183$ – основание натурального логарифма.

В качестве аргумента системной функции можно использовать алгебраическое выражение и даже системную функцию, например:

*Sin(x**3 – y*Cos(y)).*

Запись математических формул

Покажем на примерах (табл. 7.1).

Таблица 7.1

Формула	Запись на Фортране
$x_1 = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$	*x(1) = (–b + SQRT(b**2 – 4.*a*c))/(2.*a)*
$z_{i,j} = e^{\sin x_j} - \cos^3 y^2$	*z(i,j) = exp(Sin(x(j))) – Cos(y(i)**2)**3*

8. Реализация простейших программ на Fortran PowerStation

1. Подготовьте текст программы на бумаге, придумайте имя файла Вашей программы английскими буквами. Пусть это будет, например, *RUTA.*

2. В Вашем компьютере создайте папку, в которой будете сохранять программы на Фортране.

3. Выполните вход в программу: *MSDEV => BIN => MSDEV.EXE* или щелкните на экране пиктограмму *FORTRAN* или *MSDEV.*

Сверху вы увидите кнопки меню, показанные в табл. 8.1.

Таблица 8.1

Вид панели инструментов в среде Fortran PowerStation

FILE	*EDIT*	*VIEV*	*INSERT*	*BUILD*	*TOOLS*	*WINDOW*	*HELP*
Файл	Редакти-ровать	Просмотр	Вставить	Создать, построить	Инстру-менты	Окно	Помощь

4. Откройте новый файл: *FILE => New Text file => OK.* Откроется чистая страница.

5. Напечатайте текст программы.

```
Program Ruta; PRINT 11
11 FORMAT(1x, 'Programma Ruta')
PRINT 12;     12 FORMAT(1x, 3x, 'i',11x, 'y', 11x, 'z'); end
```

6. Сохраните текст: *Save as* => откройте свой раздел для сохранения =>сохранить.

7. Выполните трансляцию: преобразование исходного текста в машинную программу.

7.1. Выполните компиляцию: *Compile Ruta.f90* => да => да. Внизу появится отчет о результатах *компиляции* (первый этап трансляции):

Ruta.obj – 0 error(s), 0 warning(s). Сформирован объектный модуль *Ruta.obj*, который является «полуфабрикатом» машинной программы: каждая часть этого «полуфабриката» записана с нулевого адреса, выполнена проверка соблюдения правил записи операторов. Результат этой проверки: *0 error(s)* – ноль ошибок, *0 warning(s)* – ноль замечаний.

7.2. *Build Ruta.exe => OK.* Внизу появится отчет о завершении трансляции: *Ruta.exe – 0 error(s), 0 warning(s)*. С помощью программы-сборщика сформирована рабочая машинная программа *Ruta.exe*. Ошибок и замечаний по формальной логике программы нет.

8. Счет: *Execute Ruta.exe* ⇒ *OK*.

Сразу же появится черный экран с результатами счета.

Programma Ruta
i y z
Press any key to continue

Эти результаты вы **можете скопировать** и вставить в документ Word следующим образом:

1. Переведите курсор на синюю полосу и щелкните правой кнопкой мыши.
2. «Изменить».
3. «Выделить все».
4. Переведите курсор на синюю полосу и щелкните правой кнопкой мыши.
5. «Изменить».
6. «Копировать».
7. Переведите курсор в документ Word и в нужном месте вставьте: «Правка» => «Вставить». Вставить можно и в текстовый файл Fortran PowerStation.

Если Вам надо скопировать ***только часть результатов***, то выполните следующее:

1. Переведите курсор на синюю полосу и щелкните правой кнопкой мыши.
2. «Изменить».
3. «**Пометить**».

На черном фоне Ваших результатов в верхнем левом углу будет мигать белый квадрат. С его помощью выделите тот фрагмент результатов, который надо скопировать; получите белый прямоугольник.

4. Переведите курсор на синюю полосу и щелкните правой кнопкой мыши.
5. «Изменить».
6. «Копировать».
7. Переведите курсор в документ Word и в нужном месте вставьте: «Правка» => «Вставить». Вставить можно и в текстовый файл Fortran PowerStation. Тогда Вы сможете в одном документе

отразить результаты вычислений по нескольким задачам или нескольких вычислений по одной программе.

9. Понятие алгоритма

Единого «истинного» определения понятия «алгоритм» нет, есть множество определений. Вот лишь **некоторые из них**.

В старой трактовке алгоритм – это точный набор инструкций, описывающих последовательность действий ***исполнителя*** для достижения результата решения задачи ***за конечное время***.

Ранее часто писали «алгорифм», сейчас такое написание используется редко. Часто в качестве исполнителя выступает некоторый ***механизм*** (компьютер, токарный станок, швейная машина), понятие алгоритма необязательно относится к компьютерным программам. Так, например, четко описанный рецепт приготовления блюда также является алгоритмом, в таком случае исполнителем является человек.

*«**Алгоритм** – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает **пятью важными чертами: конечность**, **определенность, ввод**, **вывод, эффективность***» (Д.Э. Кнут).

«**Алгоритм** – это всякая система вычислений, выполняемых по строго определенным правилам, которая ***после какого-либо числа шагов*** заведомо приводит к решению поставленной задачи» (А. Колмогоров).

*«**Алгоритм** – это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату»* (А. Марков).

*«**Алгоритм** – точное предписание о выполнении в определенном порядке некоторой системы операций, ведущих к решению всех задач данного типа»* (Философский словарь / под ред. М.М. Розенталя).

*«**Алгоритм** – строго детерминированная последовательность действий, описывающая процесс преобразования объекта из начального состояния в конечное, записанная с помощью понятных исполнителю команд»* (Н.Д. Угринович, учебник «Информатика и информационные технологии»).

Здесь приведены те определения, которые, скорее всего, больше подходят к решению задач проектирования строительных объектов. Не все задачи можно решить за определенное число шагов или за определенное время. Во многих задачах вычисления прекращаются по достижении условий сходимости. Важно, чтобы алгоритм оказался *сходящимся за приемлемое время*.

Различные определения алгоритма в явной или неявной форме содержат **следующий ряд общих требований.**

Детерминированность – определенность, алгоритм *выдает один и тот же результат (ответ) для одних и тех же исходных данных.*

Понятность – алгоритм для исполнителя должен включать только те команды, которые ему (исполнителю) доступны, которые входят в его систему команд.

Завершаемость (конечность) – при корректно заданных исходных данных алгоритм должен завершить работу и выдавать результат за *конечное число шагов. Примечание:* лучше «за приемлимое число шагов».

Массовость – алгоритм должен быть применим к разным наборам исходных данных.

Результативность – завершение алгоритма определенными результатами. Алгоритм *содержит ошибки*, если приводит к получению неправильных результатов либо не дает результатов вовсе. Алгоритм *не содержит ошибок*, если он дает правильные результаты *для любых допустимых исходных данных.*

Виды алгоритмов

Особую роль выполняют **прикладные алгоритмы**, предназначенные для решения определенных прикладных задач. Для разработки алгоритмов и программ используется ***алгоритмизация*** – процесс систематического составления алгоритмов для решения поставленных прикладных задач. ***Алгоритмизация*** считается *обязательным этапом* в процессе разработки программ и решения задач на ЭВМ. Именно для прикладных алгоритмов и программ принципиально важны детерминированность, результативность и массовость, а также правильность результатов решения поставленных задач.

Форма алгоритмов

Алгоритм может быть записан ***словами*** и изображен ***схематически.*** Обычно *сначала* (на уровне идеи) алгоритм описывается *словами*, но по мере приближения к реализации он обретает все более формальные очертания и формулировку на языке, понятном исполнителю (*машинный код*).

Например, для описания алгоритма применяются *блок-схемы*.

Эффективность алгоритмов

Для каждой задачи может существовать множество алгоритмов, приводящих к цели. Увеличение эффективности алгоритмов составляет одну из задач современной информатики.

В данном пособии рассматриваются *четыре вида алгоритмов*: *линейные, разветвляющиеся, циклические, модульной структуры*. Алгоритм может быть одновременно циклическим и разветвляющимся.

10. Реализация алгоритмов линейной структуры

В линейном вычислительном алгоритме все вычисления выполняются последовательно от начала к окончанию, порядок вычислений не изменяется, алгоритм содержит конечное число шагов.

Задача 10.1. Вычисления по простейшей формуле $C = A + B \cdot D$

Схема программы	***Текст программы на Фортране***
Фиксация названия задачи	*Program Formula1*
1. Вывод на экран названия задачи	*1 print *,' Formula1 '*
2. Ввод с клавиатуры трех чисел с именами *A, B, D*.	*2 Print *,'Vvedite A,B,D' Read(*,*)A,B,D*
3. Вычисление $C = A + B \cdot D$.	*3 C= A + B*D*
4. Бесформатный вывод на экран числового значения *C*.	*4 Print *,'C=', C*
5. Останов.	*5 Pause; Stop; End*

В тексте программы в начале строк (не всех) поставлены **метки.** Это позволяет легче сопоставлять схему программы и текст программы. Наличие меток не влияет на ход трансляции программы, реализующей линейный алгоритм.

Задача 10.2. Программирование решения системы двух линейных уравнений способом определителей **(по формулам Крамера)**

$$\begin{cases} a_1 \cdot x + b_1 \cdot y + c_1 = 0; \\ a_2 \cdot x + b_2 \cdot y + c_2 = 0. \end{cases}$$

Исходные данные: a_1, b_1, c_1, a_2, b_2, c_2. ***Требуется*** вычислить x, y.

Схема программы

1. Вывод сведений о программе.
2. Ввод из области DATA значений a_1, b_1, c_1, a_2, b_2, c_2 и вывод их на экран.
3. Вычисление определителей по формуле Крамера.

$$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = \boxed{a_1 \cdot b_2 - a_2 \cdot b_1}; \; D_1 = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_1 \end{vmatrix} = \boxed{c_1 \cdot b_2 - c_2 \cdot b_1};$$

$$D_2 = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_1 \end{vmatrix} = \boxed{a_1 \cdot c_2 - a_2 \cdot c_1}.$$

4. Вычисление x, y: $x = -d_1/d$; $y = -d_2/d$.
5. Вывод на экран x, y.
6. Контроль.

Вычисление $y_1 = a_1 \cdot x + b_1 \cdot y + c_1$; $y_2 = a_2 \cdot x + b_2 \cdot y + c_2$.

Вывод на экран y_1, y_2. Конец.

y_1, y_2 – невязки уравнений; в случае правильного решения значения y_1, y_2 будут достаточно малыми по модулю, близкими к нулю.

Правильность вычислений проверяется визуально.

По составленному алгоритму подготовлена программа, в которой предусмотрен вывод промежуточных результатов.

Текст программы

```
Program ALGUR2
1  Print *,'ALGUR2';
2  continue
DATA  a1,b1,c1/10., 4., 20./;      print *,' a1,b1,c1 =', a1,b1,c1
DATA  a2,b2,c2/4., –10., –20./;    print *,' a2,b2,c2 =', a2,b2,c2
3  d = a1*b2 - a2*b1;     d1=c1*b2 - c2*b1;
 d2 = a1*c2 - a2*c1;     Print *,'3 d,d1,d2= ', d,d1,d2;
4  x = – d1/d;     y = – d2/d;
Print *,'x,y=', x,y;
5  Print *,'5 KONTROL';      y1=a1*x+b1*y-c1; y2=a2*x+b2*y-c2
Print *,'y1,y2=',y1,y2
STOP;                    End
```

Результаты вычислений

```
ALGUR2
 a1,b1,c1 =   10.000000          4.000000       20.000000
 a2,b2,c2 =   4.000000          -10.000000      -20.000000
3  d,d1,d2 =  -116.000000   -120.000000      -280.000000
x,y=       1.034483        2.413793
5 KONTROL
y1,y2=  -4.768372E-07    0.000000E+00
```

Эту же задачу можно решить методом Гаусса.

Разделим первую строку на *a*1 и умножим на *a*2, затем вычтем из строки 2 преобразованную строку 1:

*a = a2 – a1/a1*a2 = 0;* *b = b2 – b1/a1*a2;*
*c = c2 – c1/a1*a2.*

Коэффициент при *x* окажется равным нулю.

Получили одно уравнение *b*y + c = 0*, отсюда:

$y = –c/b;$ $x = (–b1*y – c1)/a1.$

Полужирным шрифтом выделены расчетные формулы.

Задача 10.3. Вычисление параметров прямоугольного треугольника

Изображение треугольника *abc* дано на рис. 10.1.

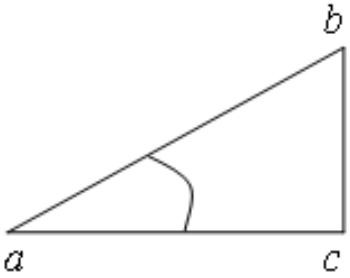


Рис. 10.1

Дано: *ac*, *bc*; требуется вычислить *ab*, синус и косинус угла *a*, площадь *S*. Ввиду простоты задачи сразу приводим ***текст программы на Фортране***.

```
Program TREUGOL1
PRINT *,'TREUGOL1'
PRINT *,'VVOD AC,BC';    READ(*,*)AC,BC;
AC=SQRT(AC**2+BC**2); SINA=BC/AB;          COSA=AC/AB;
S=BC*AC/2.;
PRINT *,'AC,SINA,COSA,S=', AC,SINA,COSA,S;   STOP;  END
```

В данном случае *SINA* – это не синус известного угла *A*, а значение синуса, вычисляемое по формуле, поэтому ее следует рассматривать как простую переменную и нельзя писать *SIN(A)* так как *SIN* – имя функции, а выражение в скобках – аргумент, значение угла в радианах.

Задача 10.4. Вычисление части площади треугольника (рис. 10.2)

В прямоугольном треугольнике даны два катета *ac, bc*. Требуется вычислить площадь *Sade* треугольника *ade*.

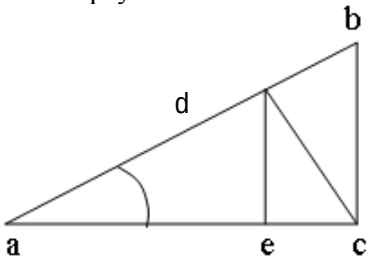


Рис. 10.2

Текст программы на Фортране

```
Program  Formula1;
PRINT *,'VVEDITE ab,bc';
READ(*,*)ab,bc
Ab=SQRT(ac**2+bc**2);
SINa=bc/ab;          COSa=ac/ab
dc=ac*SINa;          ad=ac*COSa;
de=ad*SINa;   ae=ad*COSa;  Sade=de*ae/2;
PRINT *,'ab=',ab,';   SINa,COSa=', SINa,COSa
PRINT *,'dc,de=', dc,de;      PRINT *,'ad,ae,Sade=',ad,ae,Sade;
STOP;        End
```

Смысл вычислений понятен.

Задача 10.5. Вычисление геометрических параметров трапеции

Схема трапеции с дополнительными построениями приведена на рис. 10.3. Заданными являются *h*1, *h*2, *L*, *u*. Требуется вычислить длину перпендикуляра *R*.

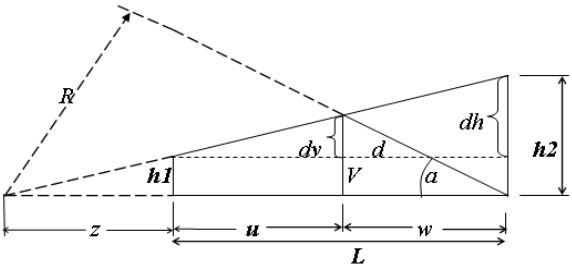


Рис. 10.3

Текст программы на Фортране

```
Program FERMA1;                    Фиксируем имя программы
REAL L                                  Описываем L как вещественное
 DATA h1,h2,L,u/1.,3.,8.,4./;      Задаем числовые значения
                                   исходных параметров фермы
70      PRINT 11, h1,h2,L,u        Выводим их на экран по формату
11 FORMAT(1x, 'h1,h2=', 2F6.2, 2x, 'L,u=', 2F6.2)

dh=h2-h1;          dv=dh*u/L;                  } Вычисляем длину
v=h1+dv;    w=L-u;     d=sqrt(v**2+w**2)       } наклонного отрезка d
sina=v/d;          cosa=w/d;                   синус и косинус его наклона
z=L*h1/dh;              Вычисляем z из подобия треугольников
R=(z+L)*sina            Вычисляем длину перпендикуляра, опущенного
                        из точки L на продолжение наклонного отрезка
PRINT *,'dh,dv,v=',dh,dv,v;
 PRINT *,'w,d=',w,d
PRINT *,'sina,cosa=',sina,cosa;      PRINT *,'z,R=',z,R;      End
```

Использовано подобие треугольников и способы решения прямоугольного треугольника.

Результаты вычислений

```
h1,h2=  1.00  3.00  L,u=  8.00  4.00
dh,dv,v=      2.000000      1.000000      2.000000
w,d=      4.000000      4.472136
sina,cosa=  4.472136E-01  8.944272E-01
z,R=      4.000000      5.366563
```

Проверено геометрическими измерениями.

Задача 10.6. Программирование расчета двухстержневого узла (рис. 10.4)

Дано: *U*1, *U*2, *v*1, *v*2, *P*, *Q*. Требуется вычислить внутренние силы в стержнях.

Вывод расчетных формул. Мысленно вырежем узел, внутренние силы обозначим буквами *F*1, *F*2, как показано на рис. 10.4, *б*.

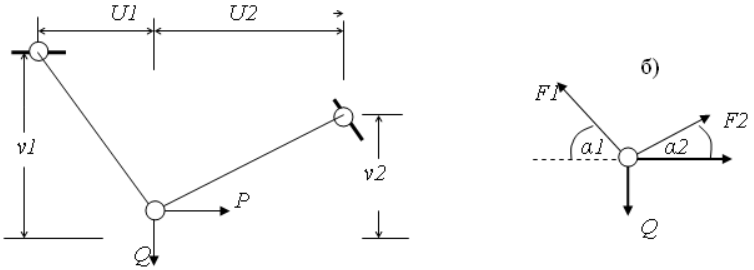


Рис. 10.4. Расчетная схема двухстержневого узла (*а*) и схема действия внешних и внутренних сил (*б*)

Спроектируем все силы на оси *X*, *Y*:

$$\sum X = -F1 \cdot \cos\alpha 1 + F2 \cdot \cos\alpha 2 + P = 0;$$
$$\sum Y = +F2 \cdot \sin\alpha 1 + F2 \cdot \sin\alpha 2 - Q = 0.$$

Решив систему уравнений относительно *F*1, *F*2, найдем внутренние силы в стержнях. При составлении программы на Фортране **используем алгоритм вычислений способом определителей из задачи 10.2.**

Схема программы

1. Начало.
2. Чтение из области *DATA U*1, *U*2, *v*1, *v*2, *P*, *Q* и вывод их на экран.
3. Вычисление геометрических параметров стержней:

$S1 = \sqrt{(U1^2 + v1^2)}$; $S2 = \sqrt{(U2^2 + v2^2)}$ – длины стержней;

$\sin 1 = v1/S1$; $\cos 1 = U1/S1$; $\sin 2 = v2/S2$; $\cos 2 = U2/S2$.

4. Определение коэффициентов при неизвестных и свободных членах системы двух уравнений:

$a1 = -\cos\alpha 1$; $b1 = \cos\alpha 2$; $c1 = -P$;

$a2 = \sin\alpha 1$; $b2 = \sin\alpha 2$; $c2 = Q$.

5. Решение знакомой системы уравнений способом определителей вычисления: $d = a1 \cdot b2 - a2 \cdot b1$; $d1 = c1 \cdot b2 - c2 \cdot b1$;

$d2 = a1 \cdot c2 - a2 \cdot c1$; $x = d1/d$; $y = d2/d$.

Контроль: $y1 = |a1 \cdot x + b1 \cdot y - c1| < 10^{-5}$? $y2 = |a2 \cdot x + b2 \cdot y - c2| < 10^{-5}$?

6. Искомые внутренние силы в стержнях $F1 = x$, $F2 = y$.

Текст программы на Фортране

```
Program Uzel2ster
1 Print *,'Uzel2ster - UZEL IZ DVUH STERGNEJ'
2 DATA U1,U2,v1,v2,P,Q/4.,4.,3.,3.,12.,0./;
Print *,' U1,U2=',U1,U2
Print *,' v1,v2=',v1,v2;                Print *,' P,Q=',P,Q;
3 Print *,'GEOMETRIJA:';
S1=SQRT(u1**2+v1**2);                   S2=SQRT(u2**2+v2**2);
Sin1 = v1/S1;       Cos1 = U1/S1;  Sin2 = v2/S2;   Cos2 = U2/S2
4 a1 = - Cos1;              b1= Cos2;           c1 = -P
a2 = Sin1;          b2 = Sin2;              c2 =  Q
5 d =a1*b2-a2*b1;  d1=c1*b2-c2*b1; d2=a1*c2-a2*c1; x=d1/d;
y=d2/d;      Print *,'d,d1,d2=', d,d1,d2;      Print *,'x,y=', x,y
Print *,'RONTROL';  y1=a1*x+b1*y-c1;      y2=a2*x+b2*y-c2;
Print *,'y1,y2=',y1,y2; Print *,'Usilija v stergniah:';  F1=x;F2=y
6 Print *,'F1,F2=',F1,F2;               Stop; End
```

Результаты вычислений по программе

```
Uzel2ster - UZEL IZ DVUH STERGNEJ
U1,U2=      4.000000       4.000000
v1,v2=      3.000000       3.000000
P,Q=     12.000000    0.000000E+00
GEOMETRIJA:
d,d1,d2=  -9.600000E-01      -7.200000       7.200000
```

x,y= 7.500000 -7.500000
KONTROL
y1,y2= -1.788139E-07 0.000000E+00
Usilija v stergniah:
F1,F2= 7.500000 -7.500000

11. Разветвляющиеся вычислительные алгоритмы

Разветвляющийся алгоритм использует специально сформулированные условия, в зависимости от выполнения которых меняется порядок вычислительного процесса. Для этого используется ***оператор IF (если***). Этот оператор имеет ***разные формы.***

Простой оператор *IF*

IF(*условие*)*выполняемый оператор*

Если условие выполнено, то выполняется предусмотренный оператор.

Если условие не выполнено, то оператор *IF* игнорируется.

Блочный оператор *IF* **с одним блоком**

IF(условие)THEN
Один оператор или блок операторов
END IF

Если условие выполнено, тогда выполняются операторы блока.

Если условие не выполнено, то оператор *IF* игнорируется.

IF – если; *THEN* – тогда; *END IF* – конец оператора *IF*;
ELSE – иначе.

Блочный оператор *IF* с **двумя блоками**

IF(*условие*) *THEN*
Один или несколько операторов (блок 1)
ELSE
Один или несколько операторов (блок 2)
END IF

В условиях можно использовать следующее.

> , <, >=, <=, ***.GT. (больше), .LT. (меньше), .EQ. (равно), .GE. (больше или равно), .LE. (меньше или равно), .NE. (не равно), .AND. (и), .OR. (или)***

На рис. 11.1–11.3 приведены блок-схемы трех операторов *IF.*

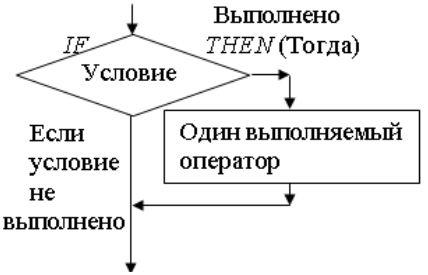


Рис. 11.1. Блок-схема простого оператора *IF*

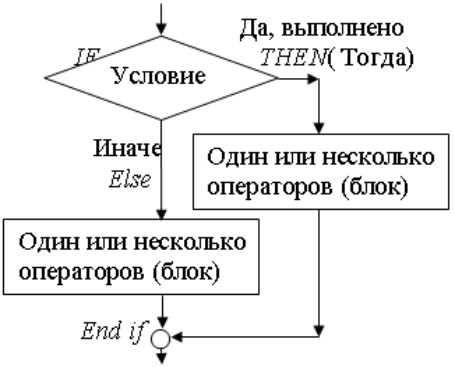


Рис. 11.2. Блок-схема блочного оператора *IF* с двумя блоками операторов

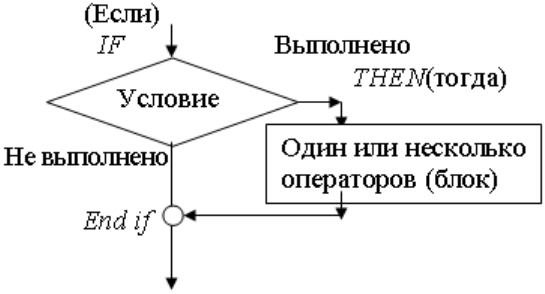


Рис. 11.3. Блок-схема блочного *Then* (тогда) оператора *IF* с одним блоком

Примеры записи операторов IF

```
IF(A>b)C = Sin(x)
IF(A.GT.b)C = Sin(x)
```

Два варианта записи одного выражения: если *A > B*, тогда вычислить *C = Sin(x)*, в противном случае оператор *IF* пропустить

```
IF(YB.GT.YA.AND.YB.GT.YC)PRI
NT *,'XB, YB = ', XB, YB
END IF
```

Если *YB > YA* и *YB > YC*, тогда вывести на печать *XB*, *YB*, в противном случае оператор *IF* пропустить

```
IF(YB>YA.AND.YB>YC)
THEN R= SQRT(A**2+b**2);
F= YA*YB
ELSE
R = 7777.;  F = YA – YB
END IF
```

Если *YB > YA* и *YB > YC*, тогда вычислить значения $R=\sqrt{(A^2+B^2)}$, *F = YA*YB*, иначе *R = 7777*, *F=YA–YB*

Задача 11.1. Вычисления по формуле, содержащей условия

$$\begin{cases} y = -0{,}5 \cdot x + 2, \text{ если } x < 4, \\ y = (x-4)^2, \text{ если } x \geq 4, \end{cases}$$

x вводится с клавиатуры.

Блок-схема программы приведена на рис. 11.4. Здесь предусмотрено использование блочного оператора *IF* с двумя блоками операторов. Эти блоки выделены полужирным шрифтом.

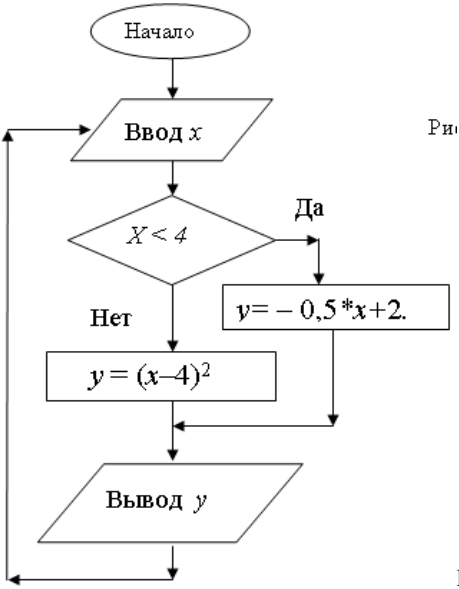


Рис. 11.4. Блок-схема программы

Текст программы

```
Program  Formula3
   10  PRINT *,'VVEDI X';    READ(*,*)x
IF(X < 4)then
y= – 0.5*x+2.
else;
y=(x – 4)**2;
end if
PRINT *,'y =', y
goto 10;      END
```

Задача 11.2. Нахождение наибольшей из простых переменных

Заданными считаются значения четырех простых переменных $B1$, $B2$, $B3$, $B4$. Требуется найти наибольшее из них.

Пусть R – значение искомого наибольшего из чисел, а k – его номер. Тогда алгоритм решения поставленной задачи выглядит следующим образом.

1. $R = B1$, $k=1$.
2. Если $B2 > R$, то $R = B2$, $k=2$, иначе (если $B2 \le R$) п. 2 пропустить.
3. Если $B3 > R$, то $R = B3$, $k=3$, иначе п. 3 пропустить.
4. Если $B4 > R$, то $R = B4$, $k=4$, иначе п. 4 пропустить.

Результат: R, k.

Этот алгоритм можно представить в виде блок-схемы, изображенной на рис. 11.5.

Текст программы

```
program RAZVETVL1
REAL B1,B2,B3,B4
DATA B1,B2,B3,B4/1.,-2.,0.5,11./; PRINT *,'B1,B2,B3,B4=',B1,B2,B3,B4
PRINT *,'B1,B2,B3,B4=',B1,B2,B3,B4;        R=B1; K=1
IF(B2.GT.R)THEN;  R=B2; K=2 ;ENDIF
IF(B3.GT.R)THEN;  R=B3;K=3;  ENDIF
IF(B4.GT.R)THEN;  R=B4;K=4;  ENDIF
PRINT *,'R,K=',R,K;           END
```

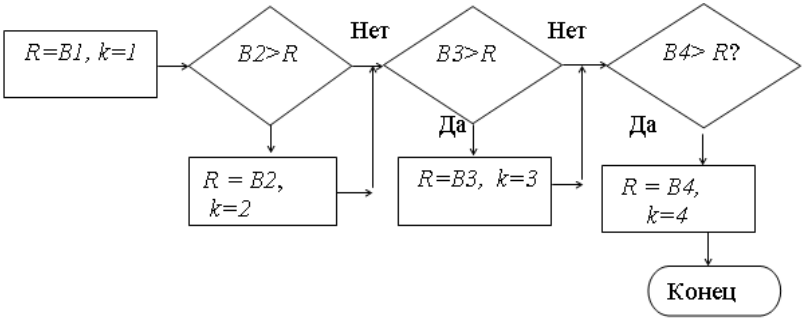


Рис. 11.5. Блок-схема алгоритма нахождения
наибольшего из значений четырех простых переменных

Задача 11.3. Организовать многократные вычисления значений функции по формуле с проверкой знаменателя на близость к нулю

В качестве примера возьмем формулу $y = (a \cdot x + b)/(c \cdot x - d)$.

Сравнивать следует абсолютное значение знаменателя с малым числом *epsilon*, которое необходимо ввести вместе с остальными исходными данными. При реальных вычислениях при делении на число, не равное нулю, но достаточно малое, результат деления может оказаться настолько большим, что решение потеряет смысл. На рис. 11.6 приведена блок-схема программы решения этой задачи.

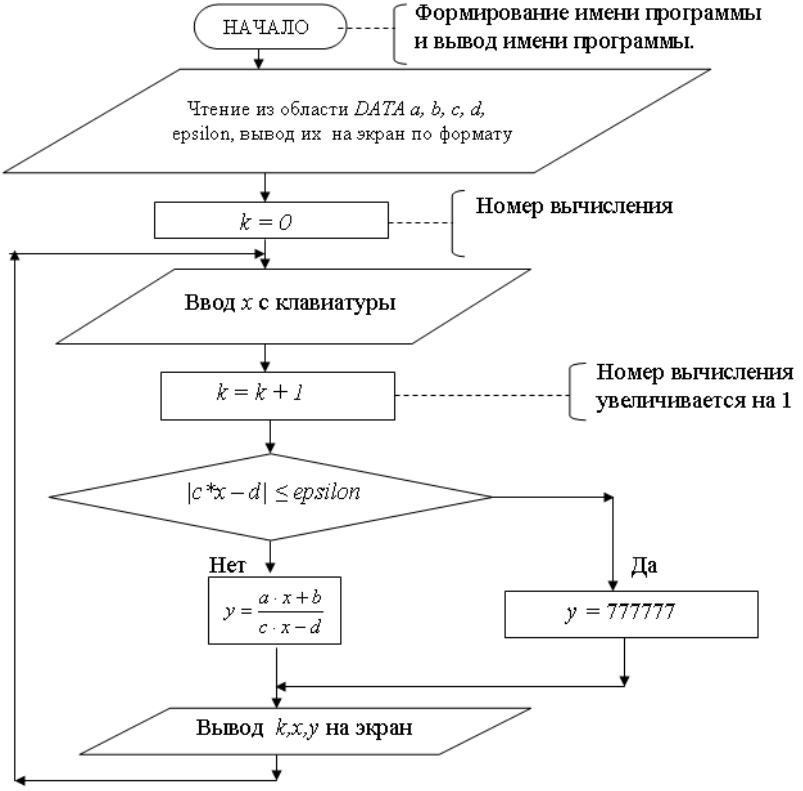


Рис. 11.6. Блок-схема многократных вычислений по формуле
$y = (a \cdot x + b)/(c \cdot x - d)$

Текст программы на Фортране

```
 Program FUNKCIJA
1 Print *,'FUNKCIJA s proverkoj znamenatelja na nol'
2 DATA a, b, c, d, eps /2.,3., 4., 5., 1.E–10/
Print 21, 'a, b=', a, b, ' c, d =', c, d,
' eps =',eps
21 FORMAT(1X, A5, 2F6.1, A8, 2F6.1, A7, 1pE9.2)
3 k = 0
4 PRINT *,'VVEDITE X'
  READ(*,*)X
5 K = K +1
6 IF(ABS(c*x – d)<= eps)then
7  y = 777777.
     else
8  y = (a*x + b)/(c*x – d)
end if
FUNKCIJA
9  print 71, 'k,x,y=', k,x,y
71 FORMAT(1x, A7, i3, F6.1, 1pE11.2)
   GOTO 4
STOP;     END
```

Здесь *A*5 – «для текстовой константы *'a, b='* выделить 5 символов». Аналогично

Проведя небольшое численное исследование, мы установили, что **это нелинейная разрывная функция**, имеющая разрыв между $x = 1{,}24$ и $x = 1{,}26$

12. **Организация циклов при помощи операторов** IF

Задача 12.1. Табуляция функции с заданным шагом изменения значения x

Дано $y = \sin x$, $x \in [xn, xk]$, шаг dx. Требуется организовать вычисление и вывод на экран k, x, y, где k – номер расчетного узла оси x.

Те формулы, которые указаны в *прямоугольных рамках*, являются **телом цикла,** содержащим **переменную цикла** *x. В ромбиках* приводятся **условия продолжения цикла или условия выхода из цикла**. Как видим, счетчик циклов может находиться до и после тела цикла.

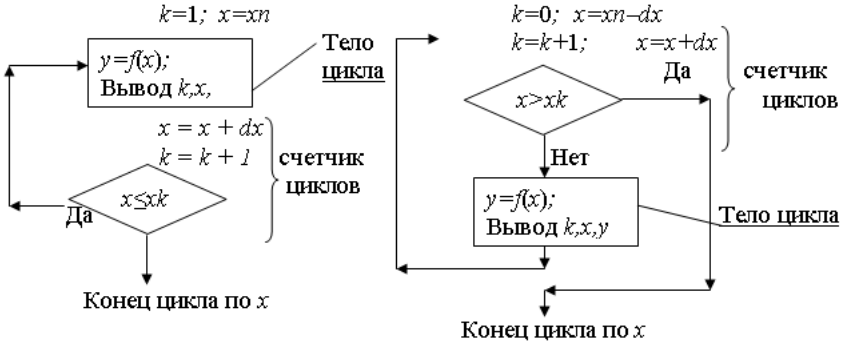


Рис. 12.1. Два варианта блок-схемы:
а – счетчик циклов и оператор *IF* расположены ниже тела цикла;
б – счетчик циклов и оператор *IF* расположены выше тела цикла

Текст программы, реализующий сначала вариант 1, затем вариант 2

```
PROGRAM CIKLISIF1;
PRINT 11,'XN,XK,DX=',XN,XK,DX;
DATA XN,XK,DX/-1.,+0.5,0.5/
11 FORMAT(1X,A9,3F4.1)
PRINT *,'..K......X......Y   VARIANT 1'
1  k=1;  x=xn
2  Y= SIN(x);; PRINT 12,k,x,y;
12 FORMAT(1X,I3,2E12.4)
3  x = x + dx ;  k = k + 1
4 IF(x<=xk)GOTO 2;       PAUSE
5PRINT              *,'..K.........X.........Y
VARIANT 2'
6 k=0;    x=xn-dx;          7    k=k+1;
x=x+dx
 8 IF(X>XK)GOTO 10
 9  Y= SIN(x); PRINT 12,k,x,y;
  GOTO 7;
 10 STOP;          END
```

Здесь точки символизируют пробелы.

Результаты вычислений

```
XN,XK,DX=-1.0.  .5
..K.......X.........Y  VARIANT 1   1
-.1000E+01  -.8415E+00
  2  -.5000E+00  -.4794E+00
  3   .0000E+00   .0000E+00
  4   .5000E+00   .4794E+00
Pause - Please enter a blank line
..K.........X........Y   VARIANT 2
  1  -.1000E+01  -.8415E+00
  2  -.5000E+00  -.4794E+00
  4   .5000E+00   .4794E+00
```

Задача 12.2. Табуляция функции с разбиением заданного интервала на N–1 отрезков

Дано $y = \sin x$, $x \in [xn, xk]$, в N узлах. Требуется организовать вычисление и вывод на экран k, x, y, где k – номер расчетного узла оси x. На рис. 12.2–12.3 показана схема разбиения оси x.

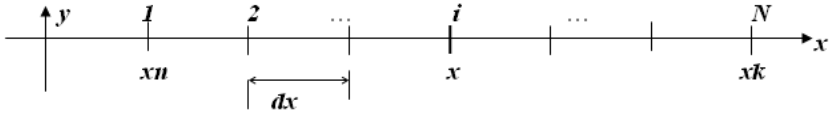


Рис. 12.2. Схема разбиения оси x

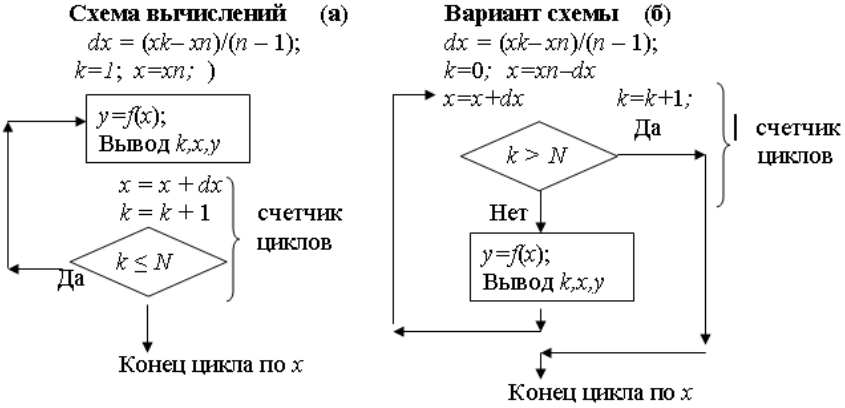


Рис. 12.3. Два варианта блок-схемы алгоритма задачи 12.2:
а – счетчик цикла ниже тела цикла; *б* – счетчик цикла перед телом цикла

Вычисления по программе, составленной в соответствии с этими алгоритмами, дали такие же результаты, что и по предыдущей программе.

Задача 12.3. Вычисление суммы элементов натурального ряда чисел

Дан натуральный ряд чисел 1, 2, 3 и т.д. и два целых числа *S*1, *S*2, причем *S*2 > *S*1.

Требуется организовать вычисление суммы ряда чисел от *S*1 до *S*2.

Схема алгоритма решения задачи

1 I=S1; Номер числа совпадает с самим числом *S*1
SUM=S1; Начальное значение суммы равно самому числу
2 I=I+1; SUM=SUM+I; Приращение номера числа и суммы
Если I < S2, то перейти к п. 2 Условие продолжения цикла
Вывод SUM

Текст программы

```
PROGRAM CIKLISIF2;
INTEGER S1,S2;          DATA S1,S2/5,10/;
PRINT *,'CIKL1SFI2  SUMMA RIADA  CELIH   S1,S2=',S1,S2
1 I=S1;      SUM=S1;        2 I=I+1;   SUM=SUM+I;
PRINT *,I,SUM;      !отладочный вывод промежуточных результатов
IF(I.LE.S2)GOTO 2;          PRINT *,'SUM=',SUM;        END
```

Задача 12.4. Вычисление координат узлов многоэтажной рамы

Для двухпролетной многоэтажной рамы **дано:** пролеты *L*1, *L*2, высота этажа *h*, число этажей *N*. Требуется сформировать массивы координат узлов (рис. 12.4).

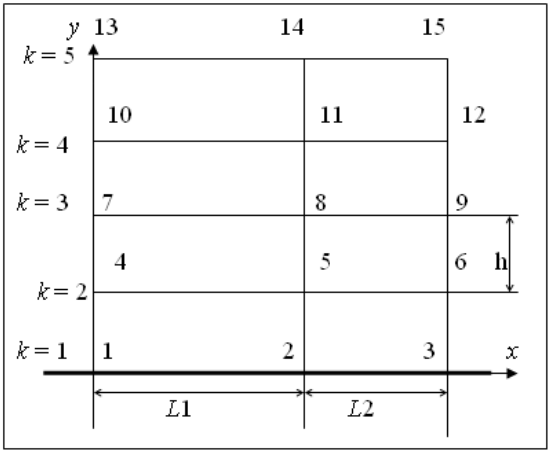


Рис. 12.4. Схема рамы

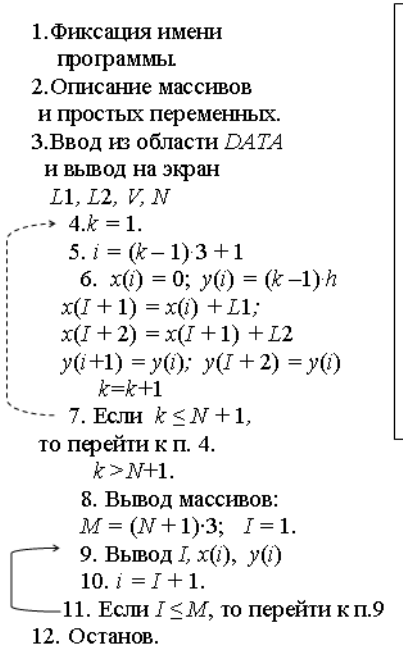
Выведем формулу определения номера 1-го в ряду узла по номеру этажа: $I = (k - 1)\cdot 3 + 1$.

Проверка: $k = 4$, $I = (4 - 1)\cdot 3 + 1 = 10$.

Координаты узлов:

$x(i) = 0{,}0;$
$y(i) = (k - 1)\cdot h.$

Схема программы



Результаты вычислений к задаче 12.4

L1,L2= 12.000000 6.000000
V= 5.000000 *N=* 4

1	0.000000E+00	0.000000E+00
2	12.000000	0.000000E+00
3	18.000000	0.000000E+00
4	0.000000E+00	5.000000
5	12.000000	5.000000
6	18.000000	5.000000
7	0.000000E+00	10.000000
8	12.000000	10.000000
9	18.000000	10.000000
10	0.000000E+00	15.000000
11	12.000000	15.000000
12	18.000000	15.000000
13	0.000000E+00	20.000000
14	*12.000000*	*20.000000*
15	*18.000000*	*20.000000*

Читателю предлагается **самостоятельно подготовить текст программы.**

Задача 12.5. Вычисление скалярного произведения двух векторов

Даны два одномерных массива $A(N)$, $B(N)$. Требуется организовать вычисление $R = A \cdot B$. Расчетная формула $R = \sum A(i) * B(i), i = 1, ..., N$.

Схема алгоритма

1. i=0; R=0.0.
2. i=i+1; R=R+ $A(i) \cdot B(i)$.
3. Если $i < N$, то перейти к п. 2.
4. Вывод на экран R.

Результаты вычислений по программе
CIKLISIF3 - RIAD S1 ... S2
A= 2.0 2.0 2.0 2.0
B= 5.0 5.0 5.0 5.0
R= 40.000000
Press any key to continue

Текст программы

```
PROGRAM CIKLISIF3;
REAL A(10), B(10)
PRINT *,'CIKLISIF3  RIAD S1 ... S2'
DATA N,  (A(I),I=1,4)/4,2.,2.,2.,2./ (B(I),I=1,4)/5.,5.,5.,5./
PRINT 11,'A=',(A(I),I=1,N);      PRINT 11,'B=',(B(I),I=1,N)
11 FORMAT(1X,A2,20F5.1);                  1 R=0.0;    I=0
2 I=I+1;      R=R+A(I)*B(I)
3  IF(I<N)GOTO 2; 4 PRINT *,'R=',SQRT(R);         END
```

Задача 12.6. Вычисление среднеквадратичного значения элементов матрицы A(m×n)

Задана вещественная матрица $A(M \times N)$. Требуется организовать вычисление

$$R = \sqrt{\sum_{i=1...M,\ j=1,...,N} A(i, j)^2}\ .$$

Алгоритм вычислений	***Комментарии***
1. $S = 0$; $i = 1$.	Первоначальное значение суммы, назначаем номер строки.
2. $j = 1$.	Назначаем номер первого элемента в строке j.
3. $S = S + A(i, j)^2$.	К имеющемуся значению суммы добавим квадрат очередного элемента $A(i, j)$.
4. $j = j + 1$.	Шаг вдоль j-й строки матрицы.
Алгоритм вычислений	***Комментарии***
5. Если $j \le N$, то перейти к п. 3.	Если учтены еще не все элементы j-й строки, то переходим к п. 3, иначе, т.е. если $j > N$, переходим к п. 6.
6. $i = i + 1$.	Переходим к следующей строке.

7. Если $i \le M$, то перейти к п. 2.

$i > M$

8. $R = \sqrt{S}$; вывод R.

Если обработаны еще не все строки, то переходим к началу новой строки – к п. 2, иначе, т.е. если обработаны уже все строки, переходим к п. 8.

Здесь мы расположили счетчик цикла после тела цикла.

Текст программы

```
PROGRAM CIKLISIF4;    REAL A(5,5);  PRINT *,'CIKLISIF4'
DATA M,N/3,  4/
DATA (A(1,J),J=1,4)/2.,2.,2.,2./
DATA (A(2,J),J=1,4)/2.,2.,2.,2./
DATA (A(3,J),J=1,4)/2.,2.,2.,2./ ;
   PRINT 11,'MATRICA  A(', M,  '*', N,  ')'
11 FORMAT(1X, A10,  I2, A1, I2, A1)
   DO I=1,M;
     PRINT 12,(A(I,J),J=1,N);  ;
   12 FORMAT(1X,20F5.1);    ENDDO
1 SUM=0;  I=1;
2  J=1;
     3  SUM=SUM+A(I,J)**2
     4  J=J+1;
           5 IF(J<=N)GOTO 3
        6 I=I+1
        7 IF(I<=M)GOTO 2
R=SQRT(SUM);     PRINT *,'R=',R;    END
```

Ввод матрицы из области *DATA* и вывод ее на экран по формату с фиксированной десятичной точкой в виде таблицы с оглавлением. Длина каждого числа 5 знаков, один знак после десятичной точки

Результаты вычислений: R = 6.92820313.

А теперь ***расположим счетчик циклов перед телом циклов.***

На рис. 12.5 приведена блок-схема программы *CIKLISIF5*, текст программы справа от блок-схемы.

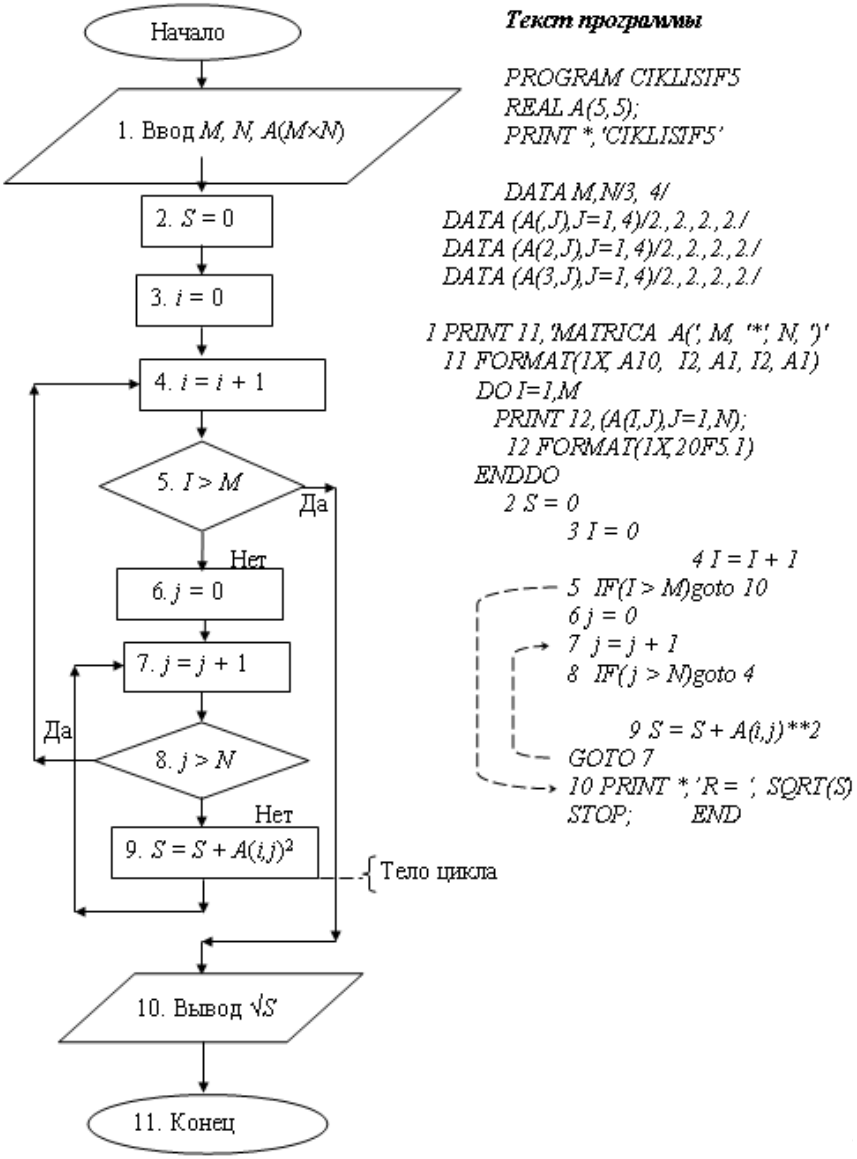


Рис. 12.5. Блок-схема программы и текст программы

13. Оператор цикла DO и оператор Continue

Используются в качестве более краткой формы записи операторов при организации циклов. Ниже приведены две схемы записи оператора *DO* (рис. 13.1).

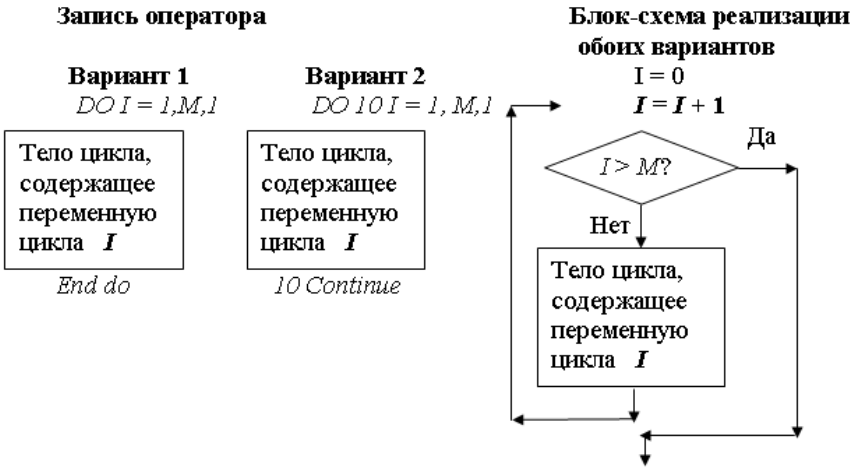


Рис. 13.1. Вариант 1 и вариант 2 записи оператора *DO*

Смысл 1-го варианта

Выполнить все операторы, расположенные между операторами *DO* и *End do*, при изменении переменной цикла от 1 до *M* с шагом 1.

Примечание. Величина шага может быть равна *L*, а не 1.

Смысл 2-го варианта

Выполнить все операторы, расположенные между оператором цикла *DO* и оператором продолжения *10 Continue* при изменении переменной цикла *I* от 1 до *M* с шагом 1.

Если переменная цикла целого типа изменяется с шагом 1, то шаг изменения ее можно не указывать. Но на начальной стадии освоения этого оператора шаг изменения переменной лучше указывать всегда. *Шаг изменения переменной вещественного типа следует указывать всегда.*

Ниже приведен вариант 3 записи оператора *DO* с переменной вещественного типа и вариант 4 с двумя переменными целого типа.

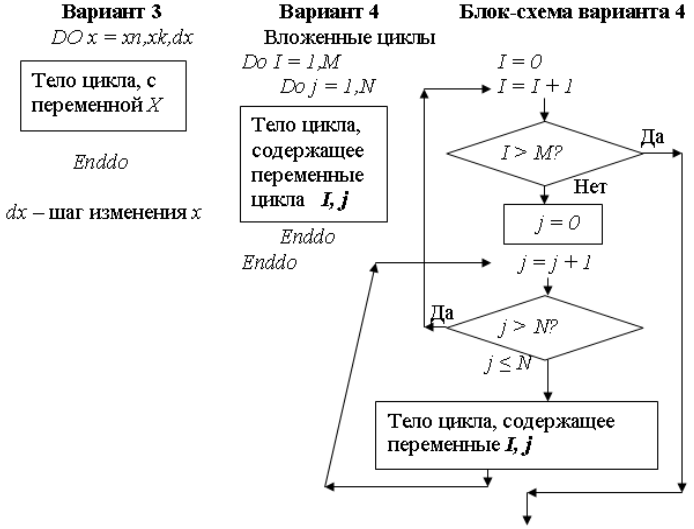


Рис. 13.2. Вариант 3 и вариант 4 записи оператора *DO*

Теперь запишем текст программы задачи 12.6 с использованием оператора *DO*.

```
PROGRAM CIKLISDO6;  REAL A(5,5);  PRINT *,'CIKLISDO6'
DATA M,N/3, 4/
DATA (A(1,J),J=1,4)/2.,2.,2.,2./
DATA (A(2,J),J=1,4)/2.,2.,2.,2./
DATA (A(3,J),J=1,4)/2.,2.,2.,2./ ;
  1   PRINT 11,'MATRICA  A(', M, '*', N, ')'
11 FORMAT(1X, A10,  I2, A1, I2, A1)
    DO I=1,M;
      PRINT 12,(A(I,J),J=1,N)
    12 FORMAT(1X,20F5.1);    ENDDO
SUM=0.;
 DO I = 1,M
   DO J = 1, N
        SUM=SUM+A(I,J)
   ENDDO
 ENDDO
 2 R = SQRT(SUM);       WRITE(*,21) 'R = ',R
 21 FORMAT(1x, A2, E12.4);          STOP;       END
```

Ввод матрицы из области *DATA* и вывод ее на экран по формату с фиксированной десятичной точкой в виде таблицы с оглавлением. Длина каждого числа 5 знаков, один знак после десятичной точки.

Машинная программа, полученная после трансляции этой фортран-программы, будет организована так же, как предыдущая.

Как видим, текст программы оказался значительно короче.

14. Ввод/вывод одномерных массивов

14.1. Ввод с клавиатуры строки X(N) и бесформатный вывод ее на экран

N – число элементов.

Пусть необходимо организовать ввод массива из *N* = 3 элементов *X* = {–3., 5.5, 8.} и вывести его на экран.

Текст программы

```
PROGRAM VVOVIVMAS1;    REAL X(5)
PRINT *,'VVOD  N'          Вывод на экран подсказки
READ(*,*)N                 Команда ввода одного целого числа, которое
                           будет направлено в ячейку памяти по имени N
PRINT *,'VVOD ',N,' chisel cherez probeli'        Вывод подсказки
READ(*,*)(x(i),i=1,N)      Чтение с клавиатуры набранной строки из N чисел
                           и пересылка ее в массив X.
10  Write(*,*) 'x=', (x(i),i=1,N)   Бесформатный вывод строки X из N чисел
END
Будет выведено по стандартному формату:
X=     -3.000000       5.500000       8.000000
```

Примечание. Выражение «чтение с клавиатуры» условно. Набранная на клавиатуре строка попадает в буфер оперативной памяти и только затем пересылается по адресам. Можно **вывести весь массив, описанный оператором** *REAL*, **упрощенным способом:**

*Print *, ' X=', X*. Будет выведен полностью весь файл из 5 чисел:

```
X=     -3.000000       5.500000       8.000000   0.000000E+00
     0.000000E+00
```

Любопытно, что если назначить N = 7 и ввести 7 чисел, то и выведено будет 7 чисел, хотя длина массива X равна 5. В сложных задачах элементы массива X займут ячейки памяти следующего массива, и это приведет к ошибкам.

Для каждого числа отводится 15 позиций.

Вывод **с помощью оператора** *FORMAT:*

30 PRINT 11, 'X=',X; 11 *FORMAT(1X, A2,25F5.1)*

В операторе *FORMAT A2* предусмотрено 2 символа для текстовой константы; *25F5.1* – оставшаяся часть зоны вывода размечена для вывода до 25 чисел с фиксированной десятичной точкой.

14.2. Ввод с клавиатуры столбца X(N)

```
40 PRINT *,'VVEDITE   N';    READ(*,*)N;
41 Do i=1,N;
     PRINT *,  'VVEDITE     X',  i;
     READ(*,*)X(i);
  End do
```

14.3. Бесформатный вывод столбца X(n)

*42 PRINT *,'Massiv X:'*

*DO i=1,n; PRINT *,X(i);* в каждой строке выводится только *X(i)*

End do

Программа, реализующая ввод/вывод одномерного массива по п.14.2 и 14.3, имеет следующий вид.

```
PROGRAM VVOVIVMAS1;     REAL X(5)
PRINT *,'VVEDI  N';       READ(*,*)N
PRINT *,'VVEDI ',N,  'CHISEL MASSIVA X'
READ(*,*)(x(i),i=1,N);
  10 Write(*,*)'x=', (x(i),i=1,N)
20 PRINT *;   PRINT 11, 'X=',X;      11 FORMAT(1X, A2,25F5.1)
30 PRINT 11, 'X=',X;
40 PRINT *,'VVEDITE   N';    READ(*,*)N;
41 Do i=1,N;     PRINT *,  'VVEDITE X',  i;  READ(*,*)X(i);  End do
42   PRINT *,'Massiv X:'
 DO i=1,n;   PRINT *,X(i);   End do;  END
```

14.4. Ввод из внешнего файла № 1 строки X(N)

и вывод ее на экран по формату

Пример внешнего файла по имени *DVVOVIVMAS2.f90.*

5 N Здесь справа от числа дано пояснение, что за число
-1.5 2. 0.75 -2.5 18.0

Текст программы

```
Program VVOVIVMAS2;                  REAL X(5)
OPEN(1,FILE=' DVVOVIVMAS2.F90');     READ(1,*)N
READ(1,*)(X(j), j=1,N);              WRITE(*,11) (X(j),j=1,N)
11 FORMAT(1x,'X=',10F7.2);           STOP;          END
```

Пояснения к операторам

OPEN(1,FILE='DVVOVIVMAS2.F90') Открыть файл № 1, имя которого ' *DVVOVIVMAS2.F90*'.

Будет установлена связь между программой *VVOVIVMAS2.F90* и файлом данных ' *DVVOVIVMAS2.F90'*.

READ(1,)N*	Прочитать из файла 1 целое число и занести в ячейку памяти с именем *N*
READ(1,)(X(j), j=1,N)*	Прочитать из файла 1 строку из *N* чисел и занести в массив, имеющий имя *X*
WRITE(,11) (X(j),j=1,N)*	Вывести на экран по формату, указанному в строке с меткой 11, *N* вещественных чисел массива *X* в виде строки
11 FORMAT(1x,'X=',10F7.2)	Вывести «*X*=» затем до 10 вещественных чисел длиной 7 символов каждое с фиксированной точкой, два знака после десятичной точки

14.5. Ввод из внешнего файла столбца и вывод его на экран

Дано: столбец *F*, содержащий *N* чисел.

Пример файла исходных данных приведен справа.

Текст программы

```
Program TOP;      REAL  F(10);
Open(1,File='DTop.f90') ;      READ(1,*)N;
 DO I=1,N;        READ(1,*)F(I);        ENDDO
Print *,'MASSIV F:'
DO i=1,N;         Print *,F(i);       ENDDO
END
```

Файл данных *DTop.f90*

```
  3    N
–4.5
  2.0
  8
```

Оператор *READ(1,*)F(I)* организует чтение из файла *DTop.f90* одного числа в строке и занесение его в память по адресу *F(i)*, т.е. формирование *i*-го элемента массива *F.*

14.6. Ввод одномерного массива из области DATA и вывод в виде строки

```
PROGRAM VVOVIVMAS4; REAL D(5);     N=3
DATA (D(J),J=1,3)/1.,-2.,-0.4/;
PRINT *,'D=',(D(I),I=1,N);          END
```

Вот так вводить нельзя:

```
N=3;
DATA (D(J),J=1,N)/1.,-2.,-0.4/; PRINT *,'D=',(D(I),I=1,N); END
```

В операторе *DATA* **нельзя использовать простую переменную для назначения числа элементов массива выводимых чисел.**

Оператором *DATA* **можно вводить массив на всю объявленную длину.**

```
PROGRAM  VVOMASDATA4;  REAL D(3);
DATA D/1.,-2.,-0.4/;          PRINT *,'D=',(D(I),I=1,3);      END
```

Так тоже правильно. Вводить часть массива в этом случае нельзя. Поэтому можно описать массив *D* на длину 5, в операторе *DATA* указать 5 чисел: три из них полезные (если *N* = 3), а остальные нули.

```
PROGRAM VVOMASDATA5;        REAL C(5);      N=3;
```

```
DATA C/1.,-2.,-0.4,0.,0./;     PRINT *,'C=',(C(I),I=1,N);      END
```

Все будет просто. При необходимости можно изменить значение *N* и полезную длину массива в операторе *DATA*.

В одной программе один массив можно ввести в память с помощью оператора DATA только один раз. Пересылка данных из области DATA в ячейки памяти простых переменных и массивов происходит при трансляции программы. Изменить их значения можно только с помощью операторов присваивания (знак «=»)

Следующие записи недопустимы:

```
DATA (D(J),J=1,3)/1.,-2.,-0.4/;      PRINT *,'D=',(D(I),I=1,N);
DATA (D(J),J=1,3)/5.,24.,-8.5/;      PRINT *,'D=',(D(I),I=1,N).
```

Описанные способы ввода одномерных массивов с помощью оператора *DATA* реализованы в программе *VVODATA1.F90.*

```
PROGRAM VVODATA1;     REAL D(5), E(5), C(5); N=3
DATA (D(J),J=1,3)/1.,-2.,-0.4/;
PRINT *,'D=',(D(I),I=1,N);
DATA E/1.,-2.,-0.4,0.,0./;     PRINT *,'E=',(E(I),I=1,N);
DATA C/1.,-2.,-0.4,0.,0./;     PRINT *,'C=',(C(I),I=1,N);      END
```

Результаты работы программы *VVODATA1.F90* **приведены ниже.**

```
D=     1.000000     -2.000000   -4.000000E-01
E=     1.000000     -2.000000   -4.000000E-01
C=     1.000000     -2.000000   -4.000000E-01
```

Естественно, вводить **оператором** *DATA* **длинные массивы сложно**. Но для учебных коротких массивов использование этого оператора вполне пригодно.

14.7. Ввод с клавиатуры таблицы данных из 5 одномерных массивов

Пусть задана таблица из 5 массивов.

Номер строки *i*	*A*	*N*	*T*	*K*	*R*
1	0.01	100	33	1011	550000000
2	–0.7	12.3	4	203	44000000
3	12	–0.3	5	–45	330000

Предполагается, что все массивы имеют одинаковую длину *S*, массивы *A, N, R* – вещественные, а массивы *T, K* – целочисленные.

Требуется организовать ввод этой таблицы по строкам и вывод с оглавлением.

Текст программы

```
PROGRAM VVOTAB1
REAL A(10),N(10),R(10);
INTEGER S,T(10),K(10)
```

Простая переменная *S* и массив *T* по умолчанию вещественные, а в программе они предполагаются целыми, поэтому они должны быть описаны типом *INTEGER*

```
   PRINT *,'VVEDITE CHISLO STROK S';      READ(*,*)S
   PRINT 11,'VVEDI';  PRINT *,' I A(I),N(I),T(I),K(I),R(I)'
   11 FORMAT(1X,A19,I2,A25);
   DO I=1,S;   READ(*,*)I,A(I),N(I),T(I),K(I),R(I);      ENDDO;
   END
   PRINT *,'..I.......A.......N...T...K.....R'
DO I=1,S;                PRINT(*,12)I,A(I),N(I),T(I),K(I),R(I);  ENDDO;
   12 FORMAT(1X,I3,  2F8.2, 2I4,  1PE12.3)
END
```

Здесь в операторе *PRINT *,'..I.....* вместо пробелов поставлены точки, символизирующие пробелы в соответствии с описанием в операторе *12 FORMAT.*

Для целой простой переменной предусмотрено 3 позиции, поэтому в оглавлении поставлены две точки и буква *I*. Для двух вещественных индексированных переменных *A(i), N(i)* предусмотрено по 8 позиций, поэтому в оглавлении поставлено 7 точек и буква *A*, затем еще 7 точек и буква *N*. Для двух целых индексированных переменных *T(i), K(i)* отведено по 4 позиции, поэтому поставлено три точки и буква *K*, затем еще три точки и буква *R*. В результате названия столбцов оказываются над последней цифрой каждого столбца. Если автору программы

захочется изменить положение букв оглавления над числами столбцов, то он может убрать часть точек после буквы *I* и все оглавление сдвинется влево, и буквы оглавления окажутся над серединами чисел.

Индексированная переменная описана с плавающей десятичной точкой: 550000000 = $5.5 \cdot 10^8$ и будет выведена в виде 5.500E+08 с тремя пробелами спереди. Если эту переменную описать E12.3, то это значит, что число 550000000 = $0.55 \cdot 10^9$ и будет выведено в виде 0.550E+09 с тремя пробелами спереди.

Ввод таблицы данных с клавиатуры утомителен, поэтому ее вводить рациональнее **из внешнего файла**, а небольшие массивы можно вводить поочереди из области *DATA*, а затем выводить в виде таблицы.

14.8. Ввод из внешнего файла таблицы одномерных массивов и вывод ее на экран

Пусть необходимо организовать ввод той таблицы данных, которая приведена выше. Подготовим файл исходных данных.

3	*A*	*N*	*T*	*K*	*R*	Первая цифра 3 – число элементов в каждом массиве – значение простой переменной *s*
	0.01	100	33	101	550000000	
	–0.7	–12.3	4	203	44000000	
	12	–0.3	5	–45	3300000	

Текст программы

```
PROGRAM VVOVIBTAB2
REAL A(10),N(10),R(10);              INTEGER S,T(10),K(10)
OPEN(1,FILE='DVVOVIVTAB2.F90')
PRINT *,'VVOVIBTAB2.f90, dannije v faile DVVOVIVTAB2.F90'

READ(1,*)S                           Чтение числа строк
DO I=1,S;                            Счетчик номеров вводимых строк
READ(1,*)A(I),N(I),T(I),K(I),R(I);   Чтение i-й строки таблицы данных
ENDDO                                Переход к следующей строке
PRINT*,'TABLICA DANNIH'              } Печать оглавления таблицы
PRINT*,'..I..A(I)...N(I)..T(I)..K(I)....R(I)'
DO I=1,S;                            Счетчик номеров выводимых
```

```
                                        строк
PRINT 11,I,A(I),N(I),T(I),K(I),R(I)     Вывод i-й выводимой строки
ENDDO
          11 FORMAT(1X,  I3,  2F10.2,  2I5,  1PE11.2)
END
```

Результаты работы программы

```
VVOVIBTAB2.f90,  dannije v faile  DVVOVIVTAB2.F90
TABLICA DANNIH
..I.....A(I).....N(I)..T(I)..K(I)....R(I)
 1      .01    100.00   33  101   5.50E+08
 2     -.70     12.30    4  203   4.40E+07
 3    12.00      -.30    5  -45   3.30E+06
```

Пояснения к оператору *FORMAT*

1X – вывод начать с новой строки, этот символ используется для управления печатью.

I3 – для выводимого целого числа отводится три позиции.

2F10.2 – для двух вещественных чисел отводится по 10 позиций. Предусмотрен вывод с фиксированной десятичной точкой, два знака после десятичной точки.

2I5 – для двух целых чисел отводится по 5 позиций.

F10.2 – предусмотрен вывод вещественного числа с фиксированной десятичной точкой. Для числа отводится 10 позиций, 2 знака после десятичной точки.

1PE11.2: 1P – сдвиг десятичной точки вправо на 1 позицию.

E – вывести в форме с плавающей десятичной точкой.

11 – число позиций, отводимых для пробела, знака и числа.

2 – знака после десятичной точки.

Число –256,24 будет выведено _ _-2.56E+02. Если формат будет иметь вид E11.2, то это же число на выводе будет иметь вид -_ _-0.26E+03. Черточки символизируют пробелы.

15. Операции над одномерными массивами

Задача 15.1. Определение суммы, произведения и наибольшего элемента массива

Задан массив A из N элементов. Требуется организовать вычисления по формуле

$$s = \sum_{i=1,\ldots,N} A(i), P = \prod_{i=1,\ldots,N} A(i), R = \max\{A(1),\ldots A(N)\}.$$

Схема программы

Начало.
Формируем имя программы. Назначим размер и тип массива A.
1. Выводим название задачи.
2. Организуем чтение N, $A(N)$ из области *DATA*.
3. Выводим массив $A(N)$ на экран.
4. Назначаем начальные значения переменным:

$S=A(1)$; $P=1$; $R=A(1)$; $I = 0$.

5. Вычисляем номер следующего элемента массива A: $I = I + 1$.
6. Анализируем I на окончания цикла.
Если $I > N$, то переходим к концу – к п. 10.
7. Имеем: $I \leq N$, вычисляем $S = S + A(I)$; $P=P \cdot A(I)$.
8. Сравним $A(I)$ и R. Если $A(I) > R$, то $R = F(I)$, $NOMER = I$.
NOMER – номер наибольшего элемента в массиве.
9. Выполним переход к продолжению цикла – к п. 5.
10. Конец цикла.
Вывод на экран результатов вычислений: *S, P, R, NOMER.*
11. Останов.
Конец.

Текст программы

```
PROGRAM MASSIV1;
REAL A(10);10 PRINT *,'MASSIV1.f90: naijti suummu S, Proizvedenije P'
PRINT *,'Naibolshij element R I EGO NOMER'
20   N=5;  DATA A/1.,-2.,-0.4  ,6,5.,0,  0,0,0,  0/;
                PRINT *,'ISHODNIJE DANNIJE:'
30 PRINT 31,'A=',(A(I),I=1,N);          31 FORMAT(1X,A2,10F5.1)
```

```
40 S=A(1);  P=1;    R=A(1);          I=1
50 I=I+1;          60 IF(I>N)GOTO 100
70 S=S+A(I);          P=P*A(I);
      80 IF(A(I)>R)THEN;    R=A(I); NOMER=I;     ENDIF
90 GOTO 50
100 PRINT *,'REZULTATI:';          PRINT
1 01,'S,P,R,NOMER=',S,P,R,NOMER
101 FORMAT(1X, A12,3F8.1,I4)
110 STOP;              END
```

Задачи для самостоятельных упражнений

Дано: массив $A(n)$. Организуйте вычисления по формулам:

$R = \max(A(i+1) - A(i)),\ i = 1,\ldots,n-1$; $R = \min|A(i)|, i = 1,\ldots,n$;

$R = \sum A(i),\ i = 2,4,6,\ldots,n$; $R = \sum (A(i) \langle 0),\ i = 1,\ldots,n$.

Задача 15.2. Формирование одномерного массива из элементов другого массива

Дано: N, K, L и одномерный массив $A(N)$.

$A(1)$, $A(2)$, ..., $A(K)$, $A(K+1)$, ..., $A(j)$, ..., $A(L)$, ..., $A(N)$, причем $L > K$.

Требуется: 1) сформировать одномерный массив B из элементов $A(K)$, $A(K+1)$, ..., $A(L)$;

2) сформировать массив C из отрицательных элементов массива A.

Алгоритм формирования массива B

$j = 1$; $i = K$ — j – номер первого элемента массива B

1 $B(J) = A(i)$

$J = j + 1$; $i = i + 1$

Если $J \le L$, то перейти к п. 1

[ПРОГРАММУ НАПИШИТЕ САМОСТОЯТЕЛЬНО]

Алгоритм формирования массива C

$J = 0$ — Текущий номер элемента массива C

$I = 1, \ldots, N$

Если $A(i) < 0$, тогда

$J = J + 1$; $C(J) = A(I)$

[ПРОГРАММУ НАПИШИТЕ САМОСТОЯТЕЛЬНО]

Следующее *J*

Задача 15.3. Нахождение наименьшего элемента массива на заданном интервале

Задан массив *A*, содержащий *N* элементов, *i1*, *i2* – номер начального элемента и номер конечного элемента массива. Подготовить **программу** нахождения наименьшего элемента массива на заданном интервале.

A(1), *A*(2), *A*(*i1*)**,…,** *A*(*i*)**,…,** *A*(*i2*),…, *A*(*N*).

Схема алгоритма

*A*min=*A*(*i*1);
i= *i*1+1,…,*i*2;
Если *A*(*i*)< *A*min, то *A*min = *A*(*i*), Nom*A*min = *i;*
следующее *i*.

Текст программы

```
PROGRAM  RIJAD1;
REAL A(10);          10 PRINT *,'RIJAD1.f90 S,P,R=MAX'
20   N=5;  DATA A/1.,-2.,-0.4  ,6.,5.,0,  0,0,0,  0/;
30 PRINT 31,'A=',(A(I),I=1,N);          31 FORMAT(1X,A2,10F5.1)
I1 = 2;                I2 = 5;            Amin = A(I1);    NOMAmin = I1
DO I=I1,I2;
IF(A(I) < Amin)THEN Amin = A(I);    NOMAmin = I; ENDIF
ENDDO;      PRINT *,'Amin, NOMAmin =', Amin,NOMAmin
END
```

Задачи для упражнений

Задан массив *A*, содержащий *N* элементов, *i1*, *i2* – номер начального элемента и номер конечного элемента массива. Подготовить программу вычисления суммы и произведения элементов на заданном интервале.

Задача 15.4. Расположить элементы массива A в порядке возрастания

Схема программы

Формируем имя программы.

Описываем массив A.

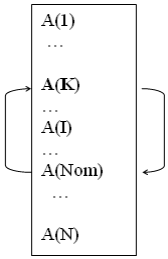
1. Формируем числовые значения элементов массива A с помощью функции sin для учебного примера. Назначим $N = 7$ – число элементов массива, начальное значение $\alpha = 0$ и шаг $d\alpha = 1.0$.

Для $I = 1,..., N$ вычислим:

$\alpha = \alpha + d\alpha$; $A(i) = \sin\alpha$.

Следующее i. Получили массив A.

2. $K = 1,..., N–1$. Перемещая k вдоль массива A, будем находить наименьший элемент среди $A(k)$, $A(k+1),...,A(N)$, а затем менять местами найденный наименьший элемент и элемент $A(K)$.



В результате наименьший элемент «всплывает наверх».

3. Amin = $A(k)$; Nom = k – назначим начальные значения.

4. Для $i = K,..., N$ выполним анализ.

Если $A(i) <$ Amin, то найден лучший элемент, и его мы запомним.

Amin = $A(i)$; Nom = i.

Следующее i.

5. Поменяем местами $A(K)$ и A(Nom):

$U=A(K)$; $V=A$(Nom); A(Nom)$=U$; $A(K)=V$.

6. Следующее K.

Текст программы

```
PROGRAM PORIADOK1;   REAL A(15);
10 N=7; AL=0; DAL=1;
  DO I=1,N; AL=AL+DAL; A(I)=SIN(AL);  END DO
PRINT 11,'A=',(A(I),I=1,N); 11 FORMAT(1X,A2,25F6.3)
20 DO K=1,N-1;              30 AMIN=A(K);          NOM=K
40   DO I=K,N;       IF(A(I).LT.AMIN)THEN;         AMIN=A(I);
      NOM=I;ENDIF;          ENDDO
50 U=A(K); V=A(NOM) ;  PRINT *,'K,NOM,U,V=',    K,NOM,U,V
   A(NOM)=U;        A(K)=V ;
  PRINT 11,'A=',(A(I),I=1,N); PAUSE
60 ENDDO
 PRINT 11,'A=',(A(I),I=1,N);          STOP;          END
```

Задача для упражнений

1. Элементы массива *A* расположите в порядке убывания.
2. Вычислите произведение отрицательных элементов массива *A*.

16. Определение геометрических характеристик фигур регулярной структуры

Задачаа 16.1. Определение координат узлов, количества и длин стержней фермы регулярной структуры

Дано: *h*1, *h*2, *L*, *N*.

Требуется организовать вычисление:

1. Координат узлов – массивы *X*(2*N*), *Y*(2*N*).
2. Количества стержней *Ks*.
3. Длин всех стержней – массив длин *D(Ks)*.

Пусть номера узлов верхнего пояса будут 1, 2, ..., *i*, *i*+1, ... ,*N*, а номера узлов нижнего пояса *N*+1, *N*+2, ..., *N*+*i*, *N*+*i*+1, ... , *N*+*N*. Тогда номера узлов первой панели будут 1, 2, *N*+1, *N*+2, а номера узлов *i*-й панели *i*, *i*+1, *N*+*i*, *N*+*i*+1. Номера стержней первой панели, не считая правой стойки, – 1, 2, 3, 4. Каждая панель содержит 4 стержня, не считая правой стойки (рис. 16.1).

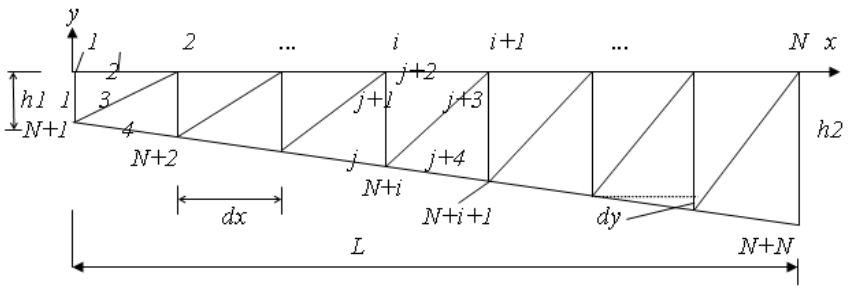


Рис. 16.1. Схема фермы:
i – номер панели и одновременно номер левого верхнего узла этой панели;
J – номер стержня нижнего пояса *i*–1 панели

Число стержней i–1 панели равно $j = (i - 1) \cdot 4$. Тогда номера стержней i-й панели, не считая правой стойки, будут j+1, j+2, j+3, j+4. А число стержней во всей ферме $Ks = (N - 1) \cdot 4 + 1$.

Длина панели $dx = L / (N - 1)$.

Координаты узлов верхнего пояса равны $x(i) = dx \cdot (i - 1)$, $y(i) = 0$, $i = 1,\ldots,N$.

Перепад высот узлов нижнего пояса в пределах панели $dy = (h2 - h1) / (n - 1)$.

Координаты узлов нижнего пояса i-й панели $y(N+i) = -h1 - dy \cdot (i-1)$. $y(N+i+1) = y(N+i) - dy$, i=1,…,N–1.

Номер стержня нижнего пояса панели № $i - 1$: $j = (i - 1) \cdot 4$.

Длины стержней i-й панели $D(j + 1) = y(i) - y(N + i)$; $D(j + 2) - dx$; $D(j + 3) = \sqrt{[dx^2 + D(j + 1)^2]}$; $D(j + 4) = \sqrt{[dx^2 + dy^2]}$.

Схема программы

1. Формирование имени программы.
2. Описание массивов, назначение наибольших их длин, описание простых переменных при необходимости.
3. Чтение из области *DATA* $h1$, $h2$, L, N и вывод их на экран по формату.
4. Вычисление:

– числа стержней фермы $Ks=(N-1)\cdot 4+1$;

– длины панели $dx = L/(N - 1)$;

– перепада высот узлов нижнего пояса в пределах панели $dy = (h2-h1)/(n-1)$;

5. Вычисление координат узлов:

– верхнего пояса: $x(i)= dx\cdot(i - 1)$, $y(i) = 0$, i=1,…,N;

– нижнего пояса $x(N + i) = x(i)$, $y(N + i)= -h1 - dy\cdot(i - 1)$, i=1,...,N.

6. Вычисление длин стержней:

$j=(i - 1)\cdot 4$, $D(j + 1) = y(i) - y(N + i)$; $D(j + 2)=dx$;

$D(j + 3)=\sqrt{[dx^2+D(j+1)^2]}$; $D(j+4)= \sqrt{[dx^2+dy^2]}$, i=1,…, N–1;

– длины последней стойки: $D(Ks) = h2$.

Текст отладочной программы

```
Program GeometFerm
Real L, x(50),y(50),D(50),L
```

```
Print *,' GeometFerm.f90 –Gtometrija fermi'; Data h1,h2,L,N/1.,3.,14.,8/
Print *,' h1,h2,L,N=', h1,h2,L,N
Ks=(N–1)*4+1;      dx=L/(N–1);     print *,'Ks,dx=', Ks,dx
dy=(h2–h1)/(N–1);  Print *,'dx,dy=', dx,dy
      print *,'koordinati uzlov'
do i=1,N;    x(i)= dx*(i–1); y(i) = 0;          ENDDO
do i=1,N;    x(N+i) = x(i);  y(N+i)= –h1–dy*(i–1); enddo
      print *,'DLINI STERGNEJ'
do i=1,N–1; j=(i–1)*4;  D(j+1)=y(i)–y(N+i);   D(j+2)=dx;   enddo
do i=1,N–1;  j=(i–1)*4;
D(j+3)=SQRT(dx**2+D(j+1)**2); D(j+4)= sqrt(dx**2+dy**2);
      enddo; D(Ks)=h2
      !Вывод таблицы результатов
Print *,' j      x      y      d'
do j=1,Ks;  Print 11,j,x(j),y(j),D(j);          Enddo
11 FORMAT(1x,i3,3F8.3);
Stop;                  End
```

На рис. 16.2 приведены схемы решеток ферм регулярной структуры для самостоятельных упражнений.

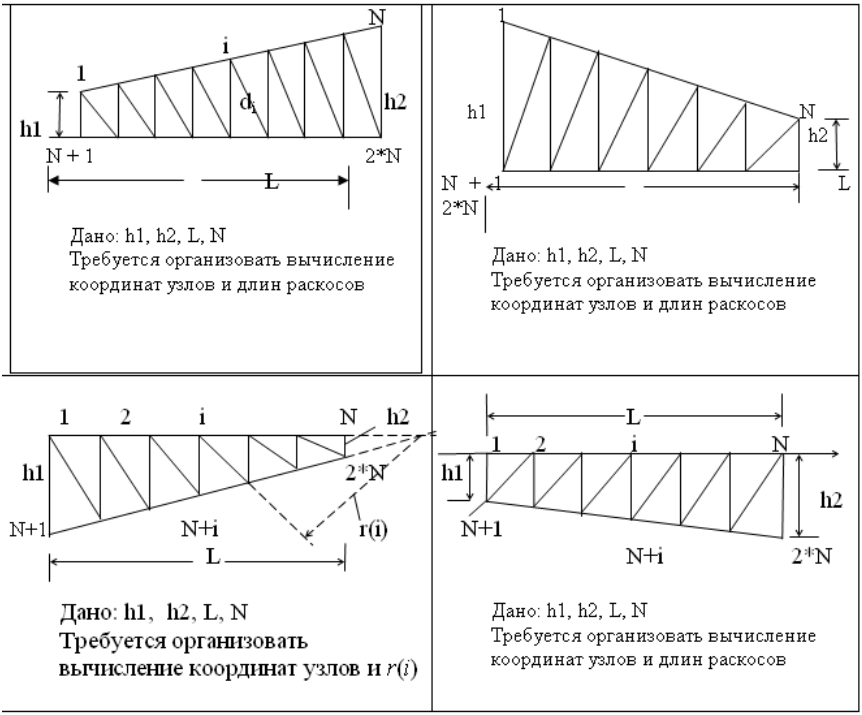


Рис. 16.2. Расчетные схемы ферм для самостоятельных упражнений

17. Функция пользователя (оператор-функция, внутренняя функция)

1**. С одним параметром:** $q(v)=\tan(v + 1)/(\sin(v) - \cos(v))$.

Пример обращения*:* $y1=(q(x + dx) - q(x - dx))/(2*dx)$.

Примечание. q – имя функции; v – аргумент, являющийся формальным параметром; $x + dx$, $x - dx$ – фактические параметры. Фактические параметры могут записываться в виде арифметических выражений. При обращении к функции программа вычислит $x + dx$ и его значение подставит в заданную функцию вместо v. В качестве формальных параметров могут быть использованы любые буквы.

2. **С двумя параметрами**: *Q*(*x*,*y*) = sin(*x*) – cos(*y*).

Обращение: *R*=*Q*(*u*,*v*)**2+*q*(*u***2,*v*+*dv*).

Примечание. *Q* – имя функции; *x*, *y* – аргументы, являющиеся формальными параметрами; *u*,*v*,*u***2*v*+*dv* – фактические параметры. Обязательно должно присутствовать два параметра для данной функции.

Функция пользователя может содержать и несколько параметров. Но в рамках данного пособия нам достаточно использовать функции до двух параметров.

3. Функция, заданная вычислительным алгоритмом.

REAL A(5)

*POL(A) = a(1)**2 + a(2)**2 + a(3)**3*

Обращение: *U = POL(f);*

здесь *f* – массив.

Примечание. *POL* – имя функции; массив *A* – формальный параметр; массив *f* – фактический параметр.

Внутреннюю функцию *можно использовать только в пределах одной программы или внутри другой функции или подпрограммы, т.е. внутри того программного модуля, в котором она сформирована.*

18. Численное исследование функции *y = f(x)*

Задача 18.1. Программирование вычисления интеграла S, формирования массивов mx, my, производных Py

Схематичное изображение графика функции *в общем случае* приведено на рис. 18.1. Интеграл вычислим как площадь между графиком функции и осью *x*. Производную – как отношение приращения ординат к приращению абсциссы.

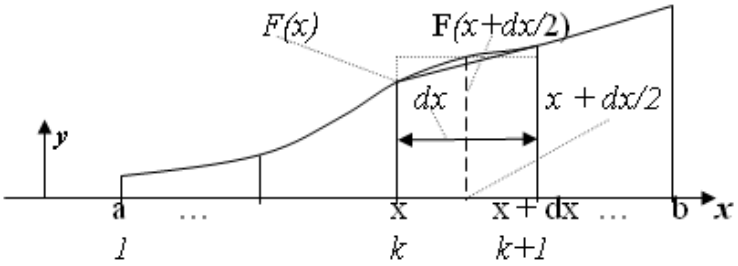


Рис. 18.1. Схематическое изображение графика функции *y = F(x)*

Дана функция $y = \sin(x)$, $x = a, ..., b$, шаг dx.
Массивы вывести в виде таблицы по формату.

```
Program Dimens
Dimension Mx(20),My(20), Py(20);  Real mx,my
Q(v)=Sin(v);                              !Оператор-функция
Data a,b,dx/0.,7.,0.5/;
Print *,'a,b,dx=',a,b,dx;  k=0; S=0;  !k – № расчетного узла на оси X
Do x=a,b,dx;
k=k+1;
y=Q(x);        mx(k)=x;         my(k)=y;
Py(k)=(Q(x+dx)-Q(x-dx))/(2*dx);     S=S+Q(x+0.5*dx)*dx;
Enddo
Write(*,*)'Integral S=',S;  N=k;
Write(*,9);              9 Format(1x,'  k',10x,'mx',10x,'my',10x,'Py');
Do k=1,N;   Print 11,k,mx(k),My(k),Py(k);              Enddo
11 Format(1x,i3,1p5E12.3);
END
```

Результаты вычислений по программе *DIMENS*

```
a,b,dx=   0.000000E+00       7.000000    5.000000E-01
Integral S=   6.602207E-01
k        mx         my              Py
1   0.000E+00   0.000E+00   9.589E-01
2   5.000E-01   4.794E-01    8.415E-01
```

```
 3   1.000E+00   8.415E-01   5.181E-01
 4   1.500E+00   9.975E-01   6.783E-02
 5   2.000E+00   9.093E-01  -3.990E-01
 6   2.500E+00   5.985E-01  -7.682E-01
 7   3.000E+00   1.411E-01  -9.493E-01
 8   3.500E+00  -3.508E-01  -8.979E-01
 9   4.000E+00  -7.568E-01  -6.267E-01
10   4.500E+00  -9.775E-01  -2.021E-01
11   5.000E+00  -9.589E-01   2.720E-01
12   5.500E+00  -7.055E-01   6.795E-01
13   6.000E+00  -2.794E-01   9.207E-01
14   6.500E+00   2.151E-01   9.364E-01
15   7.000E+00   6.570E-01   7.229E-01
```

Задача 18.2. Нахождение всех экстремумов-максимумов и минимумов

Дана функция $y = \sin(x)$, $x = a, ..., b$, в N узлах. Схема осей и расчетных узлов приведена на рис. 18.2.

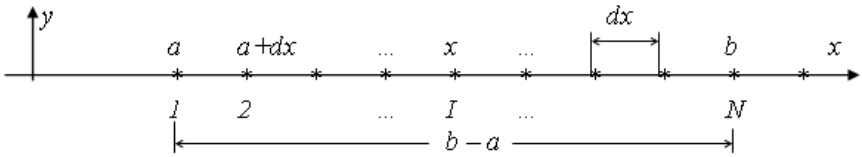


Рис. 18.2. Схема осей

Расчетные формулы: $dx = (b - a)/(N - 1)$.
Абсцисса точки i: $x = a + dx \cdot (i - 1)$.

Здесь приведен текст программы с пояснениями. Читателю предлагается с ее помощью построить блок-схему

```
Program Dimens2
Q(v)=Sin(v);
Data a,b,N/0.,7.,11/;
Print *,'a,b,N =',a,b,N;
dx = (b – a)/(N – 1); print *,'dx=',dx
```

```
PRINT *,'.....K.....X.....Y'    ! OGLAVLENIJE TABLICI
DO k = 1,N
   x = a + dx*k – 1)
PRINT *,K,X,Y
          !POISK MAXIMUMOV
          !K – НОМЕР ТЕКУЩЕГО УЗЛА
          !X – КООРДИНАТА ТЕКУЩЕЙ ТОЧКИ
IF(Q(X).GT.Q(X-DX).AND.Q(X).GT.Q(X+DX))THEN
          !ЕСЛИ Q(X)>(X-DX) И Q(X)>Q(X+DX), ТОГДА
NOM=NOM+1;  !ПОРЯДКОВЫЙ НОМЕР ОЧЕРЕДНОГО ЭКСТРЕМУМА
PRINT 21,'MAXIMUM: NOM= ' , NOM, '   K X Y=',K,X,Q(X)
          !ВЫВЕСТИ ПО ФОРМАТУ, ЗАПИСАННОМУ В СТРОКЕ С
          !МЕТКОЙ 21
          !ТЕКСТОВУЮ КОНСТАНТУ 'MAXIMUM ', ПРОСТУЮ
          !ПЕРЕМЕННУЮ NOM, ТЕКСТОВУЮ КОНСТАНТУ ' K X Y='
          !ЦЕЛУЮ ПРОСТУЮ ПЕРЕМЕННУЮ K, ВЕЩЕСТВЕННУЮ
          !ПРОСТУЮ ПЕРЕМЕННУЮ X; ЧИСЛОВОЕ ЗНАЧЕНИЕ
          !ЗАДАННОЙ ФУНКЦИИ ПРИ ТЕКУЩЕМ ЗНАЧЕНИИ
          !ПЕРЕМЕННОЙ ЦИКЛА X
ENDIF;
21 FORMAT(1X,A13,I3,A9,I3,F7.3,1PE12.4)
          !1X – ЭТО ПЕРВЫЙ ПРОБЕЛ, КОТОРЫЙ ВЫПОЛНЯЕТ ФУНКЦИЮ
          !УПРАВЛЕНИЯ ПЕЧАТЬЮ. ОН ОБЕСПЕЧИВАЕТ ПЕЧАТЬ
          !С НОВОЙ СТРОКИ
          !A13 – ЭТО ДЛИНА ТЕКСТОВОЙ КОНСТАНТЫ 'MAXIMUM: NOM='
          !I3 – ДЛЯ ВЫВОДА ЦЕЛОЙ ПЕРЕМЕННОЙ NOM ОТВОДИТСЯ
          !ТРИ ПОЗИЦИИ
          !A9 – ЭТО ДЛИНА ТЕКСТОВОЙ КОНСТАНТЫ '   K X Y='
          !I3 – ДЛЯ ВЫВОДА ЦЕЛОЙ ПРОСТОЙ ПЕРЕМЕННОЙ   K
          !ОТВОДИТСЯ ТОЖЕ 3 ПОЗИЦИИ
          !F7.3 – ВЕЩЕСТВЕННУЮ ПРОСТУЮ ПЕРЕМЕННУЮ X СЛЕДУЕТ
          !ВЫВЕСТИ С ФИКСИРОВАННОЙ  ДЕСЯТИЧНОЙ
          !ТОЧКОЙ, ПРИЧЕМ ДЛИНА ВСЕГО ЧИСЛА 7 ПОЗИЦИЙ,
          ! 3 ЗНАКА ПОСЛЕ ДЕСЯТИЧНОЙ ТОЧКИ
          !1PE12.4: 1P – СДВИГ ДЕСЯТИЧНОЙ ТОЧКИ ВПРАВО НА 1;
          ! E – КОД ЧИСЛА С ПЛАВАЮЩЕЙ ДЕСЯТИЧНОЙ ТОЧКОЙ.
          !ЭТО ЗНАЧИТ, ЧТО СЛЕДУЮЩАЯ
          !ПО ПОРЯДКУ ПРОСТАЯ ПЕРЕМЕННАЯ БУДЕТ ВЫВОДИТЬСЯ
          ! С ПЛАВАЮЩЕЙ ДЕСЯТИЧНОЙ ТОЧКОЙ
          !ПРИ ЭТОМ ОБЩАЯ ДЛИНА ЧИСЛА РАВНА 12 ПОЗИЦИЙ,
          ! А ПОСЛЕ ДЕСЯТИЧНОЙ ТОЧКИ 3 ЦИФРЫ
          !НАПРИМЕР, БУДЕТ ВЫВЕДЕНО   _-3.508E-04  =  -000.3508
          !ЕСЛИ УБРАТЬ 1P, ТО БУДЕТ ВЫВЕДЕНО  _-0.351E–03
```

```
!POISK MINIMUMOV
IF(Q(X).LT.Q(X-DX).AND.Q(X).LT.Q(X+DX))THEN
NOM=NOM+1;
PRINT 21,'MINIMUM: NOM= ' ,NOM, '  K X Y=' , K,X,Q(X)
ENDIF;  ! КОНЕЦ ЦИКЛА ПО K
        !ОДИН И ТОТ ЖЕ ФОРМАТ ИСПОЛЬЗУЕТСЯ ВО ВТОРОЙ РАЗ
ENDDO  !ПРОВЕРКА УСЛОВИЯ ПРОДОЛЖЕНИЯ ЦИКЛА
STOP;  END
```

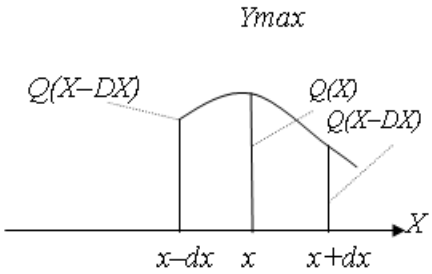


Рис. 18.3. Иллюстрация к определению экстремума-максимума

После отладки программы можно отладочную функцию заменить на другую функцию и выполнять расчеты.

Задача 18.3. Нахождение нулевых точек функции f(x)

Дана функция $y = (\sin x^2 - \cos^3 x)/(\sin x + \cos x)$.

$X \in [A, b]$ в N узлах. Требуется организовать вычисление k, x, y с анализом знаменателя на ноль и поиском нулевых точек.

Сначала выведем формулу для вычисления абсциссы нулевой точки (рис. 18.4). Вывод формулы выполним для общего случая определения положения точки пересечения с осью x прямой, проходящей через две точки.

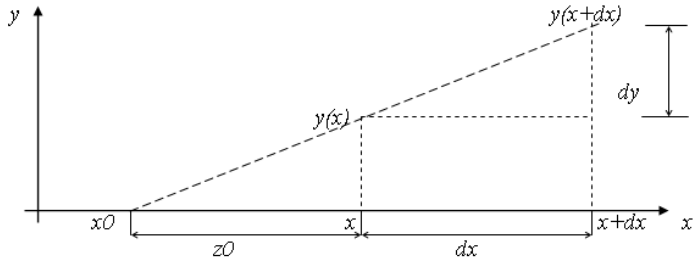


Рис.18.4. Иллюстрация к выводу формулы вычисления абсциссы $x0$

Вывод формулы для вычисления значения x:

$dy = y(x + dx) - y(x)$; => $dy/dx = y(x)/z0$; => $z0 = y(x) \cdot dx/dy$.

Отсюда $x0 = x - y(x) \cdot dx/(y(x + dx) - y(x))$.

На рис. 18.4 и 18.5 показаны общий и частный случаи применимости этой формулы.

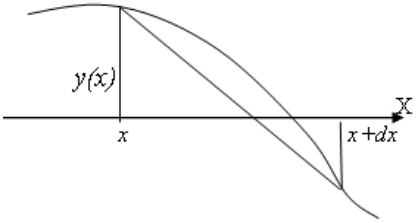


Рис. 18.5. Частный случай, когда точка пересечения прямой с осью x находится между точками x и $x + dx$ на оси x

Ниже приведена блок-схема программы этой задачи (рис. 18.6).

Исходные данные: *A, B, N, EPSILON.*

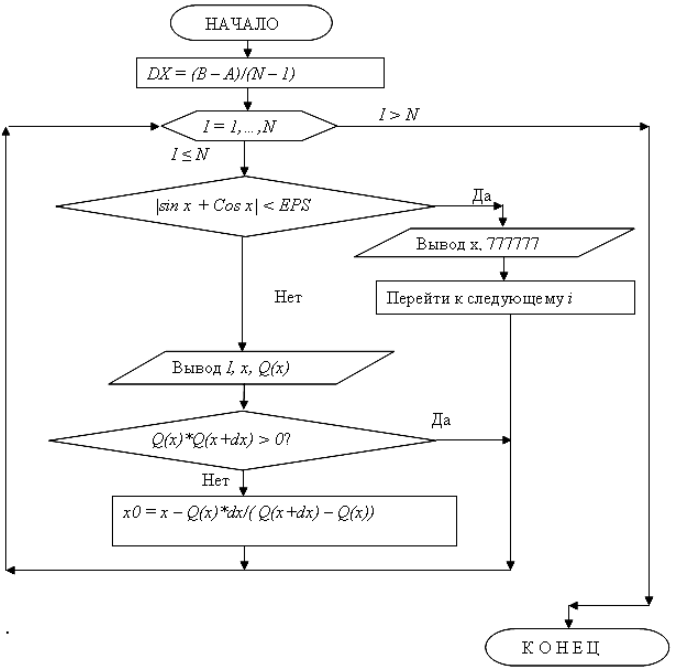


Рис. 18.6. Блок-схема алгоритма нахождения нулевых точек функции *f*(*x*)

Результаты вычислений по программе, составленной в соответствии с этой блок-схемой

```
DIMENS3  A,B,N,EP= 0.000000E+00   2.000000   11  1.000000E-10
dx= 2.000000E-01
.........I.............X.......        ......Y
        1    0.000000E+00       -1.000000
        2    2.000000E-01   -7.647129E-01
        3    4.000000E-01   -4.746866E-01
        4    6.000000E-01   -1.510289E-01
X0,Q(X0)=    6.903827E-01    3.321010E-04
        5    8.000000E-01       1.831700E-01
        6         1.000000          4.948296E-01
        7         1.200000          7.292042E-01
        8         1.400000          7.965102E-01
        9         1.600000          5.661531E-01
```

```
X0,Q(X0)=      1.766020     3.820147E-02
      10       1.800000      -1.158789E-01
      11       2.000000      -1.388491
```

19. Внешние функции и подпрограммы, подпрограмма Function и подпрограмма Subroutine

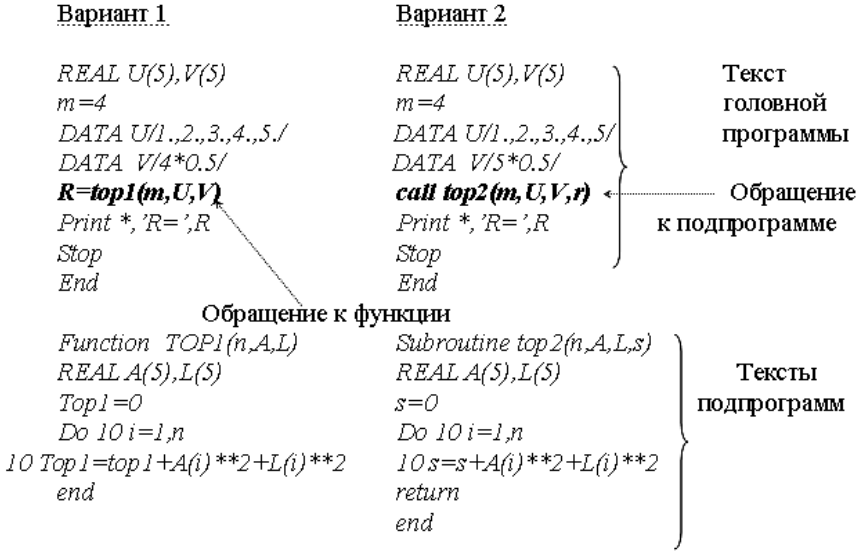
19.1. Пример использования подпрограмм Function и Subroutine

Дано: два массива U, V одинаковой длины n.
Требуется организовать вычисление значений функции

$$y = \sum[U(i)^2 + V(i)^2], \quad i = 1,\ldots,n.$$

Подготовим два варианта вычисления функции:
1) с использованием подпрограммы *Function*;
2) с использованием подпрограммы *Subroutine*.

Результатом работы подпрограммы *Function* является одна простая переменная – значение функции или алгебраического выражения. Результатом работы подпрограммы *Subroutine* может быть несколько простых переменных и массивов.



Первая строка **подпрограммы** *Function TOP1* состоит из служебного слова *Function,* имени *TOP1* и заключенного в скобки списка формальных параметров: *n* – число элементов массивов; *A, L* – имена массивов. **Имя** функции *TOP1* является одновременно и простой переменной, которая **является результатом работы подпрограммы** *Function.* Поэтому в тексте подпрограммы *Function* обязательно должен присутствовать оператор присваивания *"TOP1 ="*. Завершается подпрограмма *Function* оператором *END* или *END Function.* В качестве формальных параметров используются имена любых простых переменных и массивов, а не только тех, которые используются в головной программе.

Следом за первой строкой при необходимости приводится описание простых переменных и массивов. В нашем случае описаны массивы. При этом ***длины массивов*** в головной программе и в подпрограмме *Function* ***должны быть одинаковыми***, так как при работе программы они имеют один и тот же машинный адрес. Подпрограмма *Function* является самостоятельным программным модулем, который связан с головной программой именем подпрограммы и списком формальных параметров.

Первая строка подпрограммы *Subroutine* состоит из служебного слова *Subroutine*, имени подпрограммы *TOP2* и заключенного в скобки списка формальных параметров (*n, A, L, s*).

Список формальных параметров содержит:

n – число элементов массивов; *A, L* – имена массивов; простую переменную *S*, которая является результатом работы подпрограммы.

В общем случае результатом работы подпрограммы может быть **любое число простых переменных и массивов**. Подпрограмма *Subroutine* является самостоятельным программным модулем, который связан с головной программой именем подпрограммы и списком формальных параметров. В прежних версиях Фортрана в конце подпрограммы ставился оператор *return*, который обеспечивал формирование в рабочей программе команды возврата из подпрограммы в головную программу. Это происходит во время работы программы. В современной версии Фортрана этот оператор не применяется, но и вреда не приносит.

Задача 19.2. Использование подпрограммы решения системы двух уравнений для решения задачи о равновесии точки

Подготовим подпрограмму решения системы двух уравнений:

$a_1 \cdot x_1 + b_1 \cdot x_1 + c_1 = 0;$
$a_2 \cdot x_1 + b_2 \cdot x_2 + c_2 = 0.$

Текст подпрограммы Список формальных параметров

```
Subroutine ALGUR2(a1,b1,c1,a2,b2,c2,x1,x2)
  d=a1*b2-a2*b1; d1=c1*b2-c2*b1; d2=a1*c2-a2*c1
  x1=-d1/d; x2=-d2/d;    return;       End
Subroutine ALGUR2
```

К точке *D* приложены две заданные силы *P, Q* и две искомые силы $x1$, $x2$. Углы наклона искомых сил определяются геометрически по рис. 19.1. Исходные данные: *a, b, P, Q*.

Надо вычислить силы $x1$, $x2$.

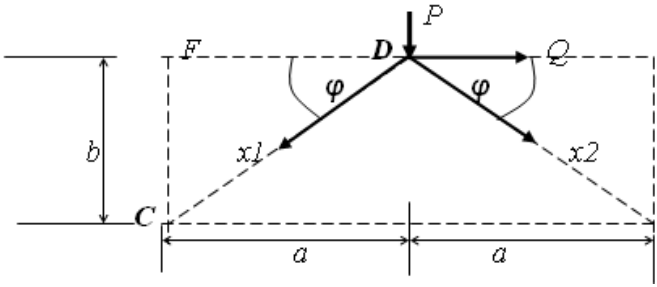


Рис. 19.1

Алгоритм вычислений. Способ проекций

1. Вычисляем геометрические параметры:

$$CD = \sqrt{a_2 + b_2}\ ; \qquad \sin\varphi = b/CD; \quad \cos\varphi = a/CD.$$

2. Приравняем к нулю сумму проекций всех сил на оси x, y; получим систему двух уравнений с двумя неизвестными, которую решим с помощью подпрограммы *ALGUR2:*

$$\sum X = -x1\cdot\cos\varphi + x2\cdot\cos\varphi + Q = 0;$$
$$\sum Y = -x1\cdot\sin\varphi - x2\cdot\sin\varphi - P = 0.$$

3. Контроль. Вычислим сумму моментов сил относительно точки F и проверим, будет ли она равна нулю:

$$\sum M_F = (x1\cdot\sin\varphi)\cdot a + (x2\cdot\sin\varphi)\cdot a + P\cdot a = 0.$$

Текст программы

```
 real mf;                          } Имя головной программы
a=4; b=3; P=6; Q=3                 } Исходные данные
Print *,'a,b,P,Q=',a,b,P,Q;        }
Print *,'SPOSOB PROEKCIJ'
```

```
cd=SQRT(a**2+b**2); SN=b/cd; cs=a/cd;          Вычисление
Print *,'cd.sn,cs=',cd,sn,cs;                  sin φ, cos φ
Call ALGUR2(-cs,cs,Q, -sn,-sn,-P, x1,x2)      В скобках список
Print *,'x1,x2=',x1,x2,                        фактических параметров
Mf=x1*sn*a+x2*sn*a+P*a; Print *,'mf=',mf;      Контроль
Write(2,*) 'kontrol: mf=', mf
End
 Subroutine ALGUR2(a1,b1,c1,a2,b2,c2,x1,x2)
 d=a1*b2-a2*b1; d1=c1*b2-c2*b1; d2=a1*c2-a2*c1
 x1=-d1/d; x2=-d2/d; return
 End          Subroutine ALGUR2
```

Результаты вычислений по программе

```
a,b,P,Q=      4.000000      3.000000      6.000000      3.000000
SPOSOB 1: SPOSOB PROEKCIJ
cd.sn,cs=     5.000000   6.000000E-01   8.000000E-01
x1,x2=     -3.125000     -6.875000   kontrol:
mf=  -3.814697E-07
```

20. Ввод/вывод двумерных массивов

Задача 20.1. Ввод с клавиатуры матрицы двумерного массива строками

```
REAL A(10,10)
PRINT*,'Vvedi M,N';READ(*,*)M,N      Ввод числа строк и числа столбцов
DO i=1,M;                            Счетчик числа строк
PRINT 11,'Naberi stroky',i
READ(*,*)(A(i,j),j=1,n);             Ввод i-й строки
END DO;                              Переход к следующей строке
11 FORMAT(1x,A13,i3,A9); END
```

A13 – для символьной константы отведено 13 символов;

i3 – для целого числа отведено 3 позиции;
A9 – для символьной константы отведено 9 символов.

Задача 20.2. Ввод с клавиатуры матрицы A(M×N) по элементам

```
REAL A(10,10)
PROGRAM VVOMATR1;        REAL A(5,5)
PRINT *,'Введите M,N';   Ввод числа строк и числа столбцов
READ(*,*)M,N
DO i=1,M;                                   Номера строк
DO j=1,N                                    Номера столбцов
WRITE(*,*)'Введите A', i, j; READ(*,*)A(i,j);   Ввод элемента A(i, j)
END DO;
ENDDO;
END
```

По этой программе вводится матрица заданного размера, но не выполняется никаких вычислений. Поэтому не следует ожидать появления на экране самой матрицы.

Задача 20.3. Ввод матрицы A(M×N) из внешнего файла

Образец внешнего файла

```
3  4   M, N                  Номер строки и номер столбца
1.0   2.   –3.    4
2.   –5.1    2.3   8.       Матрица размером M×N
1.5   6.   –15.   11
```

Эти данные записываются под именем, например, *DVVO-MATR2.f90*. В конце любой строки файла данных можно записывать пояснения, позволяющие быстро определять, что это за цифры.

Текст программы

```
PROGRAM VVOMATR2;
REAL A(5,5);
OPEN(1,FILE=' DVVOMATR2.f90')   Открытие файла № 1 под
                                 именем DVVOMATR2.f90'
READ(1,*)M,N;                   Чтение из файла 1 двух чисел и
                                занесение их в ячейки памяти
```

по именам *M,N*

```
DO i=1,M;
READ(1,*)(A(i,j),j=1,N);        Чтение из файла 1 строки i
                                 матрицы A
End do;          END
```

Задача 20.4. Ввод таблицы A(M×N), B(M) из внешнего файла и вывод на экран по формату

Файл данных

```
     3       5       M, N
5.   2.5     –8.     11.     10.
2.   –3.     15.     33.     20.    Матрица A(M×N) и столбец B(M)
-4   5.      1.      0.2     30.
```

```
PROGRAM VVOVIVtab;                REAL A(5,5);
OPEN(1,FILE=' DVVOVIVtab.f90')  Организовать связь программы
                                 с файлом данных
READ(1,*)M,N;                    Прочитать число строк
                                  и число столбцов
DO i=1,M;                       { Прочитать строку i матрицы
READ(1,*)(A(i,j),j=1,N), B(i)   { и элемента B(i)
End do;
PRINT(*,*)'ТАБЛИЦА  A(M*N), B(m):';  Вывод названия таблицы
DO i=1,M
WRITE(*,11)i,(A(i,j),j=1,N), B(i)      Вывод на экран матрицы
                                        по формату
END DO;    11 FORMAT(1x, i4, 9f8.2)
END
```

i4 – для целого числа отводится 4 позиции*;*

9f8.2 – предусмотрен вывод до 9 чисел в строке длиной 8 знаков каждое,

2 знака после десятичной точки.

Задача 20.5. Ввод из файла 1 таблицы A(m×n), B(m×t) и одновременный вывод по формату

Пример внешнего файла

```
3  4  2  m, n, t
-1.5   2.    3.   1.2    100.   50.
  5.    15    9.   7      90     60.
0.5   1.2...4.... 1     -120     70.
```

Текст программы

```
PROGRAM VVOVIVTAB1;            Real  A(5,5),b(5);  integer t
OPEN(1,FILE=' DVVOVIVTAB1.f90')
Print *,'Matrica A(M*N), B(M*t)'
read(1,*)m,n,t;
    do 10 i=1,m;
    read(1,*)(a(i,j),j=1,n),(b(I,j),j=1,t);
    write(*,11) i, (a(i,j),j=1,n),(b(I,j),j=1,t); 11 FORMAT(1x, i4, 9f8.2)
  10 continue;                  STOP;  END
```

21. Подпрограммы ввода/вывода двумерных массивов

21.1. Подпрограмма ввода двумерного массива из внешнего файла № 1

Предполагается, что файл открыт с помощью оператора *OPEN* в головной программе.

```
SUBROUTINE VVOMATR(m,n,A)     m, n, A – формальные параметры
REAL A(10,10)
READ(1,*) m, n                Чтение числа строк и числа столбцов
DO  i=1,m                     Счетчик номеров строк
READ(1,*)(A(i,j),j=1,n)       Чтение i-й строки матрицы
ENDDO;
RETURN;
 END
```

Примечание. ***Размер*** матрицы в подпрограмме ***должен совпадать*** с размером матрицы в вызывающей программе

21.2. ***Подпрограмма вывода матрицы*** *A(m×n)* ***на экран без номеров строк по формату, с плавающей десятичной точкой***

```
SUBROUTINE VIVMATR(m,n,A)     m, n, A – формальные параметры
REAL A(10,10)
do  i=1,m;
     write(*,21)(A(i,j),j=1,n);        21 format(1x,1p8E10.2)
ENDDO;     END
```

21.3. ***Подпрограмма вывода матрицы*** *A(m×n)* ***на экран с оглавлением и номерами строки***

```
SUBROUTINE VIVMATR2(m,n,A,v)    Вывод по формату
REAL A(10,10)                   с выводом имени матрицы
CHARACTER v*10          v – символьная переменная длиной
                        10 символов
WRITE(*,*)'MATRICA',v┌вслед за словом 'MATRICA' будет выведено
do  k=1,m            └10 символов содержимого переменной v
write(*,21) k, (A(k,j),j=1,n)
ENDDO;                21 format(1x, i4, 20F8.2)
END  SUBROUTINE VIVMATR2                так тоже можно
```

В вызывающей программе или в подпрограмме можно написать *CALL VIVMATR2(r,s,F,'...F(r*s):')*. В кавычках указывается содержимое символьной константы. В данном случае это имя и размер выводимой матрицы. Дополнение до 10 символов можете заполнить не точками, как в этом примере, а пробелами или черточками. Точки удобны для подсчета числа символов.

Простые переменные *r, s* ***в вызывающей программе должны быть описаны оператором*** *INTEGER*, так как по умолчанию они являются вещественными.

В результате выполнения подпрограммы будет выведено оглавление '*MATRICA ...F(r*s):*', а ниже – сама матрица. В каждой строке будет выведен номер строки *k* длиной 4 символа, а затем *n* элементов *k*-й строки по формату с фиксированной десятичной точкой. Длина каждого числа равна 8 символов, 2 знака после десятичной точки.

Результат работы этой подпрограммы по выводу матрицы *F(r*s)*:

```
MATRICA...F(R*S)
  1   -1.50    2.00    3.00    1.20
  2    5.00   15.00    9.00    7.00
  3     .50    1.20    4.00    1.00
```

22. Операции над двумерными массивами

Задан двумерный массив $A(M \times N)$. Возможно $M > N$, $N > M$, $M = N$.

Задача 22.1. Вычисление суммы всех элементов матрицы** S, **их произведения** P, **наибольшего элемента** Amax **и его адреса в матрице

Схема алгоритма

S=0; *P*=1.0; *A*max = *A*(1,1). Начальные значения переменных
Для *i*=1,..., *M;* Цикл по строкам
j=1,...,N; Цикл вдоль каждой строки
S = S + A(i,j); P = P·A(i,j).
Если *A*(*i,j*) больше *A*max, тогда
*A*max = *A*(*i,j*); *Ni=i;* *Nj=j*.
Следующее *j*.
Следующее *i*.
Вывод *S*, *P*, *A*max, *Ni*, *Nj*.

Текст программы

```
PROGRAM OPER1MATR1
REAL A(5,5);                OPEN(1,FILE='DOPER1MATR.F90')
CALL VVOMATR(M,N,A);
CALL VIVMATR2(m,n,A,'A(m*n)=...')
S=0;                P=1.0; Amax = A(1,1)
Do i=1,m;           do j=1,n;
S=S+A(i,j);         P=P* A(i,j)
IF(A(i,j)>Amax)THEN;
   Amax = A(i,j);           Ni=i;          Nj=j
```

```
ENDIF
ENDDO;    ENDDO;
     PRINT 11,'S,  P,  Amax,Ni,Nj=', S,  P,  Amax ,Ni,Nj
     11 FORMAT(1X,A19,3F9.1,2I4); END
SUBROUTINE VVOMATR(m,n,A); !   m,  n,  A – формальные па-
раметры
REAL A(5,5)
READ(1,*) m,n;
DO  i=1,m;          READ(1,*)(A(i,j),j=1,n);
ENDDO;     END   SUBROUTINE VVOMATR
  SUBROUTINE VIVMATR2(m,n,A,v);
  REAL A(5,5)
  CHARACTER v*10;          !v  символьная простая переменная
  WRITE(*,*)'MATRICA',v
  do  k=1,m;          write(*,21) k, (A(k,j),j=1,n);     ENDDO;
     21 format(1x, i4, 20F8.2);        END SUBROUTINE VIVMATR2
```

Задача 22.2. Подготовить подпрограмму решения задачи 22.1 и обратиться к ней из головной программы

```
PROGRAM OPER1MATR2
REAL A(5,5);                OPEN(1,FILE='DOPER1MATR.F90')
CALL VVOMATR(M,N,A);   CALL VIVMATR2(m,n,A,'A(m*n)=...')

     CALL SPAMAXMATR(M,N,A,S,P,Amax,Ni,Nj);  PRINT *;
PRINT 12,' V PODPROGRAMME S, P, Amax, Ni, Nj =', S, P, Amax,Ni,Nj
     12 FORMAT(1X,A19,3F9.1,2I4)
END PROGRAM OPER1MATR2

SUBROUTINE SPAmaxMATR(M,N,A,S,P,Amax,Ni,Nj)
REAL A(5,5)
       S=0;          P=1.0; Amax = A(1,1)
  Do i=1,m;      do j=1,n;      S=S+A(i,j);       P=P* A(i,j)
     IF(A(i,j)>Amax)THEN;          Amax = A(i,j);  Ni=i;   Nj=j;
     ENDIF
ENDDO;    ENDDO;
END  SUBROUTINE SPMAXMATR
```

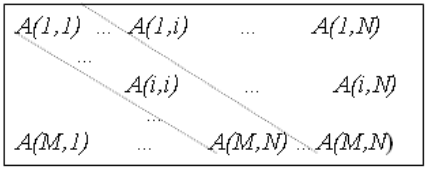
Имя подпрограммы образовано из: *S*, *P*, *A*max, *Matrica*.

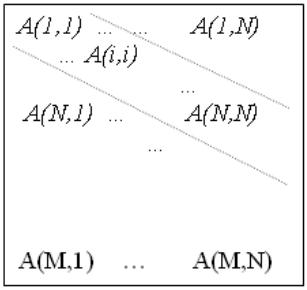
Подпрограммы ввода и вывода матрицы те же, что в задаче 22.1 и здесь не приведены.

Задача 22.3. Организовать вычисление S, P, Amax элементов главной диагонали

Дано: *M*, *N*, *A*(*M*×*N*).

Требуется вычислить *S*, *P*, *A*max, а также адрес *A*max в матрице – номер строки *Ni* и номер столбца *Nj*.





Возможны три варианта: *N* > *M*, *N* = *M*, *N* < *M*. Первый и третий варианты показаны на рисунках.

Схема алгоритма вычислений

S=0; *P*=1.0; *A*max = *A*(1,1).	Назначаются начальные значения
i = 1,…, *m*;	Перебираются строки
j=1,…, *N*.	Перебираются все элементы каждой строки
Если *i* = *j*, тогда	Если *i* = *j*, то имеем элемент главной диагонали
S = *S* + *A*(*i*,*j*);	Наращивается сумма
P = *P*·*A*(*i*,*j*).	Наращивается произведение
Если *A*(*i*,*j*) > *A*max, тогда	Выполняется сортировка
*A*max=*A*(*i*,*j*), *ni*=1, *nj*=*j*.	Запоминается наибольший элемент
Следующее *j*.	Переход к следующему элементу строки

Следующее *i*. Переход к следующей строке
Вывод *S, P, Amax, Ni, Nj*. Вывод результатов

Возможен способ движения по главной диагонали ***без использования оператора*** *DO* – с помощью оператора *IF*, анализируя каждый шаг вдоль диагонали.

Схема программы

1. $S = A(1,1)$; $P = A(1,1)$;
$Amax = A(1,1)$; $i=1$; $j=1$. } Начальные значения

2. $i = i + 1$. Шаг вниз по главной диагонали

3. Если $i>M$ или $j > n$, то перейти к окончанию вычислений – к п. 7, в противном случае п. 3 пропустить. } Проверяются условия окончания счета

4. $S = S + A(i,i)$; Наращивается сумма
$P = P \cdot A(i,i)$. Наращивается произведение

5. Если $A(i,i) > Amax$, тогда $Amax=A(i,i)$, $ni= i$, $nj=j$, } Выполняется сортировка, запоминается наибольший элемент
в противном случае п. 5 пропустить.

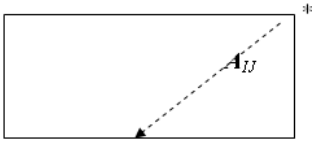
6. Перейти к следующему шагу – к п. 2.

7. Вывод *S, P, Amax, Ni, Nj* на экран.
8. Останов.

Задача 22.4. Вычисление S, P, Amax элементов побочной диагонали матрицы A(M×N), причем N > M

Схема алгоритма

В качестве начального выберем несуществующий элемент за пределами матрицы. На рисунке этот элемент показан символом «*».



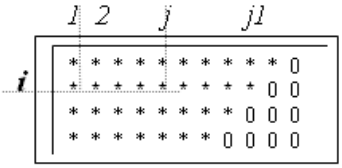
1. i=0; $j = N + 1$; Номер строки и номер столбца
$S = 0.0$; $P = 1.$; Начальные значения искомых переменных
Amax= $A(1,N)$. В качестве начального значения Amax можно принять любой элемент диагонали
2. $i = i + 1$; $j = j - 1$. Шаг вдоль диагонали

3. $S = S + A(i,j)$; $P = P \cdot A(i,j)$. Наращивание S, P
4. Если $A(i,j) >$ Amax, тогда Amax = $A(i,j)$; $Ni = i$; $Nj = j$. } Сортировка: запоминаем наибольший элемент и его адрес
5. Если $I < M$, то перейти к п. 3. Проверка условия продолжения вычислений

6. Вывод S, P, Amax, Ni, Nj на экран.

Задача 22.5. Вычисление S, P, Amax элементов ненулевого фрагмента

В прямоугольной матрице $A(M \times N)$, в которой $N > M$, выделена область, отмеченная на рисунке символами «*».



Требуется вычислить сумму элементов S этой области, произведение P и наибольший элемент Amax.

Схема алгоритма

1. $i = 1$; S=0; P=1.0; Amax = $A(1,1)$;
$j1 = N - 1$; Наибольшее значение номера столбца в i-й строке

j=0. Начальное значение номера столбца переменных
2. $j = j + 1$. Шаг вдоль строки

3. $S = S + A(i,j)$; $P = P * A(i,j)$. } вычисление S, P, Amax, ni, nj
4. Если $A(i,j) >$ Amax, тогда Amax = $A(i,j)$; $Ni = i$; $Nj = j$.

5. Если $j < j1$, то перейти к п. 2, Проверка условия продолжения

цикла по строке i

$j=j1$.

6. $j=0$; $i = i + 1$; $j1=j1-1$;

если $i \le M$, то перейти к п. 2. Переход к следующей строке

7. Вывод $S, P, Amax, Ni, Nj$ на экран.

Задача 22.6. В прямоугольной матрице $A(M\times N)$ удалить столбец k

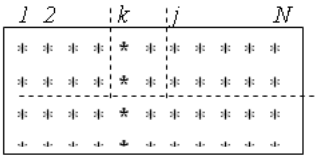


Схема алгоритма

$j = k,\ldots, N-1$;
  $i = 1,\ldots, M$.
    $B(i,j)=B(i,j+1)$.
  Следующее i.
Следующее j.

Задача 22.7. В прямоугольной матрице $A(M\times N)$ вставить столбец $V(M)$ после столбца k

Для этого надо сначала раздвинуть столбцы после столбца k, увеличить N на 1, а затем элементам столбца $k+1$ присвоить значения вставляемого столбца $V(M)$.

Схема алгоритма

$j = N,\ldots, k+1$, шаг -1;
$i = 1,\ldots,M$; $B(i,j + 1) = B(i,j)$.
  Следующее i.
Следующее j.

В результате содержимое столбцов $k+1,\ldots, N$ будет сдвинуто вправо, содержимое столбцов $k+1$ и $k+2$ окажется одинаковым

$i = 1,\ldots, M$; $B(i,k+1) = V(i)$.
Следующее i.

Элементам $k+1$-го столбца присваиваются значения элементов вставляемого столбца V

23. Решение задач линейной алгебры

Задача 23.1. Сложение векторов

Дано: $A=\{A_1\ A_2\ A_3\ A_4\}$; $B=\{B_1\ B_2\ B_3\ B_4\}$.
Вычислить: $R = A + B = \{ A_1 + B_1\ A_2 + B_2\ A_3 + B_3\ A_4 + B_4 \}$.
Схема алгоритма: $j = 1,\ldots,N;$ $R(j) = A(j) + B(j)$. Следующее j.

Задача 23.2. Скалярное произведение векторов

Дано: $A=\{A_1\ A_2\ A_3\ A_4\}$; $B=\{B_1\ B_2\ B_3\ B_4\}$.
Требуется организовать вычисление

$$R = A_1 \cdot B_1 + A_2 \cdot B_2 + A_3 \cdot B_3 + A_4 \cdot B_4 = \sum(A(i) \cdot B(i)),\ i=1,\ldots,4.$$

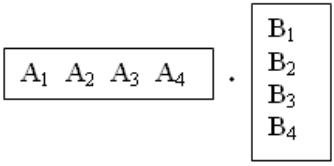
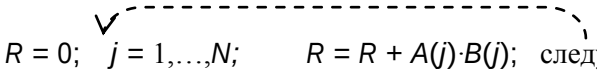


Схема алгоритма:



Задача 23.3. Сложение матриц

Дано: M, N, $A(M\times N)$, $B(M\times N)$.
Расчетная формула: $R(i,j) = A(i,j) + B(i,j)$, $i = 1,\ldots, M;$ $j=1,\ldots, N.$
Пример вычислений вручную:
A + B => R

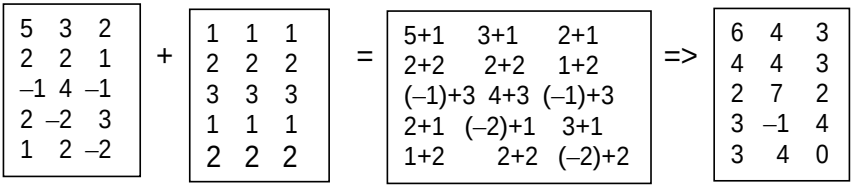
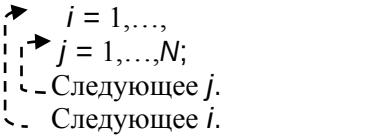


Схема алгоритма



Задача 23.4. Умножение матрицы на вектор-столбец

Дано: M, N, $A(M\times N)$, $F(N)$.

Расчетная формула: $R(i) = \sum(A(i,j)\cdot F(j))$, $j = 1,\ldots, N$; $i=1,\ldots, M$.

Пример:

$$R = A*F = \begin{bmatrix} 5 & 3 & 2 \\ 2 & 2 & 1 \\ -1 & 4 & -1 \\ 2 & -2 & 3 \\ 1 & 2 & -2 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 5*1+ & 3*2+ & 2*3 \\ 2*1+ & 2*2+ & 1*3 \\ -1*1+ & 4*2+ & (-1)*3 \\ 2*1+ & (-2)*2+ & 3*3 \\ 1*1+ & 2*2+ & (-2)*3 \end{bmatrix} = \begin{bmatrix} 17 \\ 9 \\ 4 \\ 7 \\ -1 \end{bmatrix}$$

Для каждой строки i вычисляется сумма произведений элементов этой строки и заданного столбца.

Схема программы

$i = 1,\ldots, M$;	Назначается номер строки
$S = 0.0$;	
$j = 1,\ldots, n$;	Вычисляется сумма
$S = S + A(I,j)\cdot F(j)$.	произведений элементов
Следующее j.	этой строки и столбца F
$R(i) = S$.	Эта сумма пересылается в i-й элемент результирующего вектора R
Следующее i.	Выполняется переход к следующей строке

Задача 23.5. Умножение двух матриц

Дано: M, N, $A(M\times N)$, $B(N\times T)$.

Результат: матрица $R(M\times T)$.

Расчетная формула: $R_{ik} = \sum(A(i,j)\cdot B(j,k))$, $j = 1,\ldots, N$; $i = 1,\ldots, M$; $k = 1,\ldots, T$.

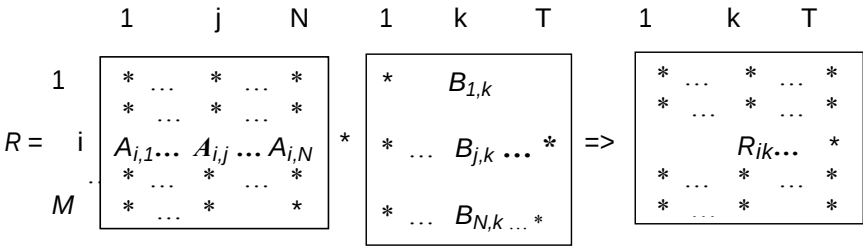
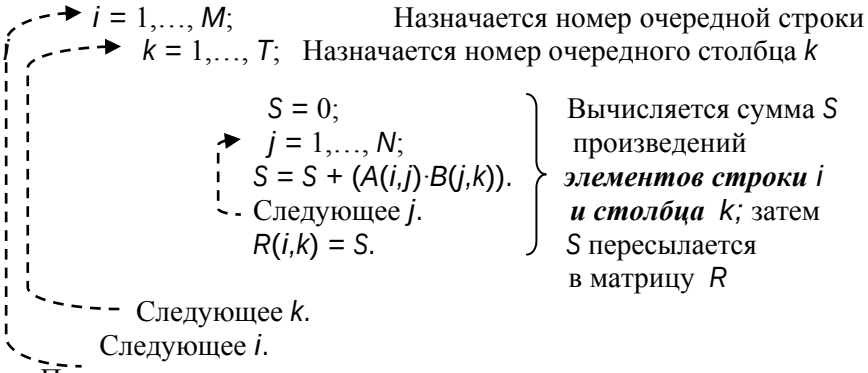


Схема программы



Пример:

$$\begin{bmatrix} 1 & 1 \\ 3 & 2 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 5 \\ 1 & 1 & 4 \end{bmatrix} = \begin{bmatrix} 1*2+1*1 & 1*2+1*1 & 1*5+1*4 \\ 3*2+2*1 & 3*2+2*1 & 3*5+2*4 \\ 2*2+1*1 & 2*2+1*1 & 2*5+1*4 \end{bmatrix} =$$

$$= \begin{bmatrix} 3 & 3 & 9 \\ 8 & 8 & 23 \\ 5 & 5 & 14 \end{bmatrix}$$

Задача 23.6. Транспонирование матрицы

Дано: M, N, $A(M \times N)$.
Результат: $AT(N \times M)$.

Каждая строка матрицы A будет установлена в матрице AT в виде столбца.

Расчетная формула: $AT(j,i) = A(i,j)$, $j=1,\ldots, N$; $i = 1,\ldots, M$.

Схема алгоритма

$i = 1,\ldots, M$;
  $j=1,\ldots, N$;
    $AT(j,i) = A(i,j)$.
  Следующее j.
Следующее i.

Пример:

$A(4*3)$

1	2	3
3	4	5
5	5	6
7	8	7

=>

$AT(3*4)$

1	3	5	7
2	4	5	8
3	5	6	7

Отладка: в таблице, приведенной ниже, показана отладка алгоритма для двух строк матрицы A.

i		$A(I,j)$	$AT(j,i)$
1		1	1
1		2	2
1		3	3
2		3	3
2		4	4
2		5	5

$i = 1,\ldots, N$; $j = 1,\ldots, N$.

Задача 23.7. Вычисление определителя квадратной матрицы методом Гаусса

Дано: N, $A(N\times N)$.

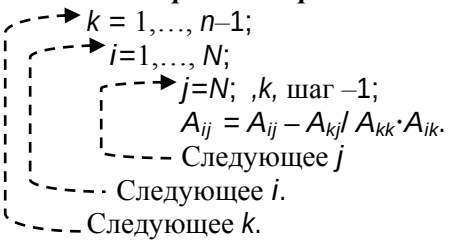
Требуется вычислить $D = det[A_{ij}]$, $i = 1,\ldots, N$; $j = 1,\ldots, N$.

Принципиальная схема вычислений

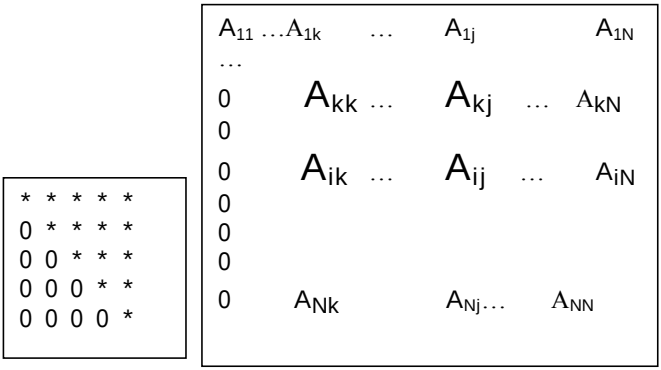
Сначала выполним *прямой ход Гаусса*, в результате которого матрица будет преобразована в *треугольную*, в верхний треугольник, включая главную диагональ. Ниже главной диагонали элементы будут равны нулю.

В соответствии с правилами линейной алгебры *величина определителя не изменится.* Но вычислить его теперь будет очень легко: достаточно вычислить *произведение элементов главной диагонали.*

Схема алгоритма. Прямой ход Гаусса



Матрица примет вид:



Вычисляем определитель как произведение элементов главной диагонали:

$D = \prod A_{ii}$, $i = 1,\ldots, N$.

Задача 23.8. Решение системы линейных алгебраических уравнений $A \cdot X + B = 0$ методом Гаусса

Дано: N, $A(N \times N)$, $B(N)$.

Результат: $X(N)$.

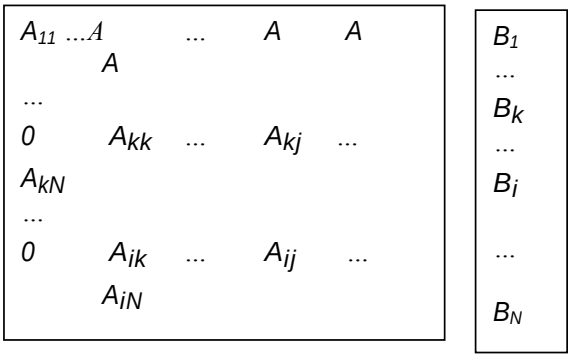
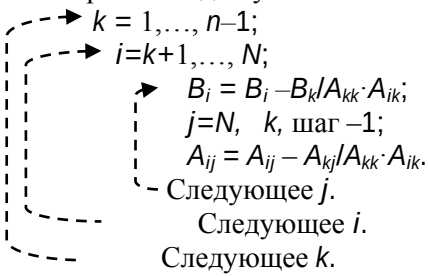


Схема алгоритма

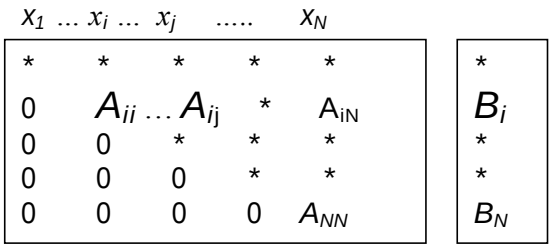
1. Сохранение данных для последующей проверки решения:

$C = A$; $D = B$.

2. Прямой ход Гаусса:



В результате прямого хода Гаусса матрица коэффициентов при неизвестных и столбец свободных членов примут вид:



3. Обратный ход Гаусса.

Нижняя строка системы уравнений содержит только один ненулевой коэффициент при неизвестном, поэтому:

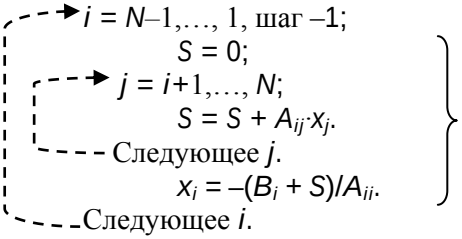
$X_N = -B_N / A_{NN}$;
i-я строка системы уравнений имеет следующий вид:

$A_{ii} \cdot x_i + A_{i,i+1} \cdot x_{i+1} + \ldots + A_{ij} \cdot x_j + \ldots + A_{iN} \cdot x_N + B_i = 0$, отсюда
$i = N-1,\ldots,1$, шаг –1.

Расчетная формула для вычисления x_i:

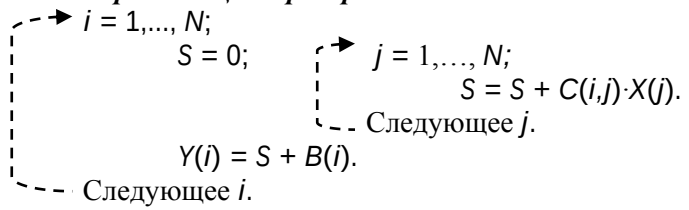
$$x_i = -[B_i + \sum(A_{ij} \cdot x_j)]/A_{ii}, \quad j = i+1, \ldots, N, \quad i = N-1, 1, \quad \text{шаг } -1.$$

Схема реализации обратного хода Гаусса



4. Проверка правильности решения:

$Yi = \sum[C(i,j) \cdot X(j)] + D(i)$,
$j = 1,\ldots, N$; $i = 1,\ldots, N$.
При правильном решении $| Y_i | \le \varepsilon$, где ε – малое число.
Схема реализации проверки



Вывод на экран $Y(i)$ для последующего анализа.

Задача 23.9. Решение системы линейных алгебраических уравнений $A \cdot X + B = 0$ методом Гаусса с выбором

ведущего элемента по матрице

Если на главной диагонали имеются нули или нули появляются при прямом ходе Гаусса, то используется перестановка строк и столбцов на каждом шаге, чтобы передвинуть наибольшие элементы на главную диагональ.

В соответствии с законами линейной алгебры перестановка строк и столбцов не приводит к изменению определителя матрицы коэффициентов при неизвестных и не изменяет результатов преобразованной таким образом системы линейных уравнений. Но полученные искомые неизвестные окажутся расположенными не в исходном порядке, их очередность будет нарушена. Излагаемый ниже алгоритм решает эту проблему.

Дано: n, $A(n \times n)$, $B(n)$.

Алгоритм вычислений

1. $A(n \times n) => C(n \times n)$; Пересылаем $A(n \times n)$ и $B(n)$ в дополнительные
   $B(n) => D(n)$. массивы для последующей проверки
2. Формируем два одномерных массив номеров строк:
   $NJ(n) = (1, 2, 3, \ldots, n)$;
   $NI(n) = \{1, 2, 3, \ldots, n)$. Массив номеров столбцов

В памяти компьютера сформированы массивы, которые удобно изобразить в виде табл. 23.1.

Таблица 23.1

NJ:	*Nj(1)*	*Nj(2)*		*Nj(k)*		*Nj (s)*		*Nj(n)*	
1	2	3	4	5	6	7	8	9	10
	1	*2*		*k*		*s*		*N*	
1	*a(1,1)*	*a(1,2)*	...	*a(1,k)*	...	*a(1,s)*	...	*a(1,n)*	*b(1)*
2	*a(2,1)*	*a(2,2)*		*a(2,k)*		*a(2,s)*		*a(2,n)*	*b(2)*
...									
k	*a(r,1)*	*a(k,2)*		*a(k,k)*		*a(k,s)*		*a(k,n)*	*b(k)*
...									
i	*a(I,1)*	*a(I,2)*		*a(i,k)*		*a(I,s)*		*a(I,n)*	*b(i)*
...									

Окончание табл. 23.1

1	2	3	4	5	6	7	8	9	10
r	*a(r,1)*	*a(r,2)*		*a(r,k)*		*a(r,s)*		*a(r,n)*	*b(r)*
...									
n	*a(n,1)*	*a(n,2)*		*a(n,k)*		*a(n,s)*		*a(n,n)*	*b(n)*

3. Выполняем **прямой ход Гаусса**:
$K = 1,\ldots, N-1$.
Назначаем номер ведущего элемента на главной диагонали матрицы.

$Q = A(N, N)$; $r = k$; $s = k$;
$i = k,\ldots,N$;
$j = k,\ldots,N$
Если $|A(i,j)| > Q$, $r = i$, $s = j$;
$Q = A(i, j)$.
Следующее j.
Следующее i.

Находим наибольший по модулю элемент фрагмента матрицы A:
$A(r,s) = \max\{A(i,j);$
$i = k,\ldots,N;\quad j = k,\ldots,N$

$j = 1,\ldots,N$;
$u = A(k,j)$; $v = A(r,j)$;
$A(k,j) = v$; $A(r,j) = u$.
Следующее j.
$u = B(k)$; $v = B(r)$; $B(k) = v$; $B(r) = u$;
$L = Ni(k)$; $M = Ni(r)$;
$Ni(k) = M$; $Ni(r) = L$.

Меняем местами:
строки k, r, **а так же**
$B(k)$, $B(r)$
$Ni(k)$, $Ni(r)$

$i = 1,\ldots,n$;
$u = A(i,k)$; $v = A(i,s)$;
$A(i,k) = v$; $A(i,s) = u$.
Следующее i
$L = Nj(k)$; $M = Nj(s)$;
$Nj(k) = M$; $Nj(s) = L$.

Меняем местами:
столбцы k, s, **а так же**
$Nj(k)$, $Nj(s)$

Выполняем *шаг прямого хода Гаусса* как в обычном алгоритме:

i=1,…, N;

$B_i = B_i - B_k/A_{kk} \cdot A_{ik}$;

j=N, k, шаг –1.

$A_{ij} = A_{ij} - A_{kj}/A_{kk} \cdot A_{ik}$.

Следующее j.

Следующее i.

Следующее k.

Переход к следующему шагу

4. Выполняем **обратный ход Гаусса,** вычисляем элементы **неупорядоченного** вектора искомых неизвестных *XX*.

$XX(n)=B(n)/A(n,n)$;

$i = n$–1,…,1, шаг –1;

SUM=0;

$j = i$+1,…, n;

$SUM = SUM + A(i,j) \cdot XX(j)$.

Следующее j.

$XX(i) = -(B(i) + SUM)/A(i, j)$.

Следующее i.

В результате имеем неупорядоченный вектор *XX(n)* искомых неизвестных и вектор соответствующих номеров

5. Формируем упорядоченный вектор *X*.

i = 1,…, N;

$k = MJ(i)$; $X(k)=XX(i)$.

Следующее i.

Данные для отладочного примера:

	Nj(n)	*XX(n)*	*X(N)*	
1	2	–8	15	1
2	5	11	–8	2
3	1	15	0.5	3
4	3	0.5	7	4
5	4	7	11	5

Пусть $i = 2$, тогда $k = Nj(2) = 5$;

$X(5) = XX(2) = 11$.

6. Проверка правильности решения системы уравнений:

$Yi = \sum[C(i,j)\cdot X(j)] + D(i)$, $j = 1,\ldots, N;\ i = 1,\ldots, N.$

При правильном решении $| Y_i | \le \varepsilon$, где ε – малое число

Схема реализации проверки

$i = 1,..., N$;

$S = 0$; → $j = 1,\ldots, N$;

$S = S + C(i,j)\cdot X(j)$.

Следующее j.

$Y(i) = S + D(i)$.

Следующее i.

Вектор значений $Y(i)$ выводится на экран для последующего визуального анализа.

24. Решение «векового» уравнения

$$y = \begin{vmatrix} a_{11}-x & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22}-x & & \\ & & \ddots & \\ a_{n1} & & \ldots & a_{nn}-x \end{vmatrix}.$$

Требуется организовать вычисление y при изменении x от xn до xk с шагом dx. Для учебных примеров ограничимся размером матрицы $n = 5$.

Задача 24.1. Формирование подпрограммы-функции вычисления определителя D методом Гаусса (функции, заданной вычислительным алгоритмом)

Дано: n, x, $A(n\times n)$.

Требуется вычесть из элементов главной диагонали x и вычислить определитель преобразованной матрицы.

Порядок вычислений

1. Пересылаем матрицу A в рабочий массив B для того, чтобы не испортить исходный массив A.
2. В матрице B исправляем элементы главной диагонали:

$$b(i,i) = b(i,i) - x, \quad i = 1,\ldots, n.$$

3. Выполняем прямой ход Гаусса над матрицей B. В результате матрица B будет преобразована в треугольную (верхний треугольник).
4. Вычисляем значение определителя как произведение элементов главной диагонали матрицы B.

```
Function D(n, x, A)
Real A(5, 5), B(5,5)
1 do i = 1,n; do j = 1,n;       b(i,j) = a(i,j);   enddo;  enddo
2 do i = 1,n; b(i,i) = b(i,i) – x;   enddo
3 do k =1,n;
     do i = k + 1,n
     do j = n,k, – 1
     b(i,j) = b(i,j) – b(k,j)/b(k,k)*b(i,k)
     enddo
     enddo
   enddo
4 D = 1;   do i = 1,n;    D = D*A(i,i);    enddo
end Function
```

$$\begin{matrix} b_{11} & b_{1k} \ldots b_{1j} \ldots & b_{1n} \\ b_{k1} \ldots & b_{kk} \ldots b_{kj} \ldots & b_{kn} \\ b_{i1} \ldots & b_{ik} \ldots & b_{ij} \ldots \quad b_{in} \\ b_{n1} \ldots & b_{nk} \ldots b_{nj} \ldots, & b_{nn} \end{matrix}$$

Задача 24.2. Формирование таблицы данных для численного исследования «векового уравнения»

Схема головной программы

Real A(5,5) – чтение из внешнего файла *n, xn, xk, dx, A*(*n*,*n*) и вывод их на экран.

k=0 – № расчетного узла на оси *x*.

X = *xn*,…, *xk*, шаг *dx*;

k = *k*+1.

Вывод *k*, *x*, *D* (*n*, *x*, *A*).

Пример файла данных

4 1.5 8 0.5	*n, xn, xk, dx*		
10	*4.5*	*0.8*	*–1.5*
4.5	*15*	*–2.*	*3*
1.4	*–2*	*20*	*–5*
3	*4*	*5*	*–25*

Следующее x

Исходные данные для численного примера приведены в рамке.

Следует иметь в виду, что **если x окажется равным $A(1,1)$**, то при вычислении $b_{ij} = b_{ij} - b_{ik}/b_{kk}b_{ik}$ ноль окажется в знаменателе и вычисления будут остановлены из-за «переполнения» ячейки памяти: число окажется настолько большим, что значение показателя степени будет нельзя разместить в отведенном месте ячейки памяти.

Иногда ноль на главной диагонали появляется в процессе реализации прямого хода Гаусса.

Задача 24.3. Нахождение нулевых точек функции

Задача решается, как для функции $y = f(x)$;

$$y = \begin{vmatrix} a_{11}-x & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn}-x \end{vmatrix}.$$

Исходные данные n, xn, xk, dx, A (рис. 24.1).

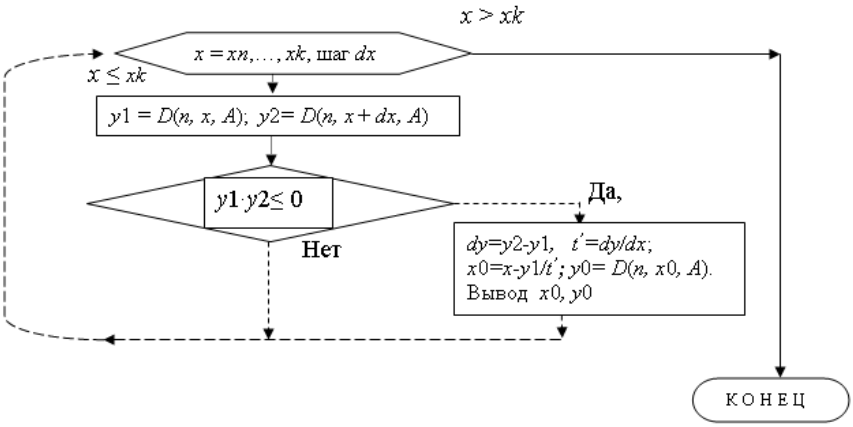


Рис. 24.1. Алгоритм решения

25. **Программирование матричной формулы**

Требуется организовать выполнение вычислений по матричной формуле $R = \boldsymbol{F}\cdot(U + D^T)$, используя подпрограммы ввода матрицы *VVOMATR*, вывода матрицы *VIVMATR2*, умножения двух матриц *UMMATR*, транспонирования *TRANS*, сложения двух матриц *SLOMATR*.

Общий план действий

1. Определим размеры матриц.

Назначим: $F(m\times n)$; => $U(n\times s)$; $DT(n\times s)$; $D(s\times n)$.

Имеем размеры матриц *m, n, s.*

2. План матричных операций:

$D(s\times n)$ => $DT(n\times s)$;

$U(n\times s) + DT(n\times s)$ => $R1(n\times s)$;

$F(m\times n)\cdot R1(n\times s)$ => $R(m\times s)$.

3. Подготовка данных для отладочного примера:

$m = 2$; $n = 3$; $s = 4$.

$$F(m\times n) = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 2 & 2 \end{bmatrix};$$

$$U(n\times s) = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 2 & 1 & 1 & -2 \end{bmatrix}; \quad D(s\times n) = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 2 & 1 \\ 2 & 2 & 1 \end{bmatrix}.$$

4. Решение отладочного примера вручную:

$$DT(n\times s)=\begin{bmatrix}-1 & 1 & -1 & 2\\ -1 & 1 & 2 & 2\\ 1 & -1 & 1 & 1\end{bmatrix};\quad R1(n\times s)=U+DT=\begin{bmatrix}-2 & 2 & -2 & 3\\ 0 & 0 & 3 & 1\\ 3 & 0 & 2 & -1\end{bmatrix}$$

$$F(m\times n)\cdot R1(n\times s)=\begin{bmatrix}1 & -1 & 1\\ -1 & 2 & 2\end{bmatrix}\cdot\begin{bmatrix}-2 & 2 & -2 & 3\\ 0 & 0 & 3 & 1\\ 3 & 0 & 2 & -1\end{bmatrix}\Rightarrow$$

$$\Rightarrow\begin{bmatrix}-2+0+3 & +2-0+0 & -2-3+2 & +3-1-1\\ +2-0+6 & -2+0+0 & +2+6+4 & -3+2-2\end{bmatrix}\Rightarrow\begin{bmatrix}+1 & +2 & -3 & +1\\ +8 & -2 & +12 & -3\end{bmatrix}$$

5. Подготовка подпрограмм.

   Ввод из внешнего файла матрицы $A(m\times n)$.

   Вывод на экран матрицы $A(m\times n)$.

   Транспонирование: $A(m\times n) => AT(n\times m)$.

   Умножение матриц: $A(m\times n)\cdot B(n\times s) => C(m\times s)$.

6. Файл исходных данных:

```
2 3              m,n
1  –1   1   F(m*n)
–1  2   2
3  4                n,  s
–1  1  –1   1   U(n*s)
 1 –1   1  –1
 2  1   1  –2
4  3              s,n
–1 –1   1   D(s*n)
 1  1  –1
–1  2   1
 2  2   1
```

7. Схема головной программы.

Имя программы: *MODUL1.*

Массивы вещественные:

F(5×5), *U*(5×5), *D*(5×5), *DT*(5×5),

*R*1(5×5), *R*(5×5).

Целые простые переменные: *m, n, s.*

7.1. Чтение из файла № 1 по подпрограмме *VVOMATR* и вывод по подпрограмме *VIVMATR2* матриц: *F*(*m*×*n*), *U*(*n*×*s*), *D*(*s*×*n*).

7.2. Транспонирование по подпрограмме *TRANS: D*(*s*×*n*) => *DT*(*n*×*s*).

Вывод по подпрограмме *VIVMATR2* матрицы *DT*(*n*×*s*).

7.3. Сложение матриц по подпрограмме *SLOMATR:*

U(*n*×*s*) + *DT*(*n*×*s*)=> *R*1(*n*×*s*).

Вывод по подпрограмме *VIVMATR2* матрицы *R*1(*n*×*s*).

7.4. Умножение матриц по подпрограмме *UMMATR:*

F(*m*×*n*)·*R*1(*n*×*s*) => *R*(*m*×*s*).

Вывод по подпрограмме *VIVMATR2* матрицы *R*(*m*×*s*).

7.5. Останов.

8. Текст головной программы:

```
      program MODUL1
      REAL F(5,5),  U(5,5), D(5,5), DT(5,5),   R1(5,5), R(5,5)
       INTEGER m,n,s     ! m, n – целые по умолчанию, их описание
                         ! как целых не обязательно
    1  call VVOMATR (m,n,F)
               call VIVMATR2(m,n,F, ' F')
               call VVOMATR(n,s,U)
call VIVMATR2(n,s,U)
call VVOMATR (s,n,D)
               call VIVMATR2 (s,n, D, ' D')
    2  call TRANS(s,n,D,DT)
               call VIVMATR2(s,n,DT, ' DT')
    3  call slomatr(n,s, U, DT, R1)
call VIVMATR2(n,s,R1, ' R1')
    4  call UMMATR(m,n,s,F,R1, R)
call VIVMATR2(m,s,R, ' R')
     5 print *,'K O N E C   S C H E T A'
STOP;                 END
SUBROUTINE VVOMATR(m,n,A)          ! m ,n, A – формальные
                                     параметры
```

```
REAL A(10,10)
READ(1,*) m,n                    ! чтение числа строк
                                   и числа столбцов
DO  i=1,m                        ! счетчик номеров
                                   строк
READ(1,*)(A(i,j),j=1,n)          ! чтение i-й строки
                                   матрицы
ENDDO;              END
```

Примечание. Размер матрицы в подпрограмме должен совпадать с размером матрицы в вызывающей программе.

Подпрограмма вывода матрицы *A(M×N)* на экран с номерами строк и с выводом оглавления матрицы:

```
SUBROUTINE VIVMATR2(m,n,A,v)          !вывод по формату
REAL A(10,10)                          !с выводом имени матрицы
CHARACTER v*10                !v – символьная переменная длиной
                               10 символов
WRITE(*,*)'MATRICA',v
DO  k=1,m
write(*,21) k, (A(k,j),j=1,n)
ENDDO;              21 format(1x, i4, 20F8.2)
END  SUBROUTINE VIVMATR2
```

Подпрограмма умножения матриц *A*(*m*×*n*)·*B*(*n*×*t*)=> *C*(*m*×*t*):

```
      SUBROUTINE UMMATR(m,n,t,a,b,c)
      integer m,n,t
      real a(5,5),b(5,5),c(5,5)
      do 40 i=1,m              ! i – номер строки матрицы A
        do 30 k=1,t            ! k – номер столбца матрицы B
          c(i,k)=0             ! c(i,k) – результат умножения строки
                               i на столбец k
          do 20 j=1,n          ! j – номер столбца матрицы A
20          c(i,k)=c(i,k)+a(i,j)*b(j,k)
30        continue
40      continue
```

```
end
```

Подпрограмма сложения матриц $A(m\times n)+B(m\times n)=>R(m\times n)$:

```
SUBROUTINE SLOMATR(M,N,A,B,R)
        real A(5,5),B(5,5),R(5,5)
DO I=1,N;   DO J=1,N;   R(I,J)=A(I,J)+B(I,J);   ENDDO;
ENDDO
END SUBROUTINE SLOMATR
```

Транспонирование матрицы

```
SUBROUTINE TRANS(M,N,A,AT)
REAL A(5,5), AT(5,5)
DO I = 1,M
  DO J = 1,N
        AT(J,I) = A(I,J)
   ENDDO
ENDDO
END
```

26. Программирование задач механики с использованием подпрограмм решения уравнений

Задача 26.1. Программирование расчета фермы путем вырезания всех узлов и решения системы уравнений

Задана расчетная схема фермы (рис. 26.1), точки приложения и направления сил.

Требуется подготовить программу вычисления усилий в опорных связях и в стержнях.

Результаты вычислений по программе *MODUL1*

```
MATRICAF(m*n)  U(
  1   1.00  -1.00   1.00
  2  -1.00   2.00   2.00
MATRICAU(n*s): D(
  1  -1.00   1.00  -1.00   1.00
  2   1.00  -1.00   1.00  -1.00
  3   2.00   1.00   1.00  -2.00
MATRICAD(s*n): DT
  1  -1.00  -1.00   1.00
  2   1.00   1.00  -1.00
  3  -1.00   2.00   1.00
  4   2.00   2.00   1.00
MATRICADT(s*n):R1
```

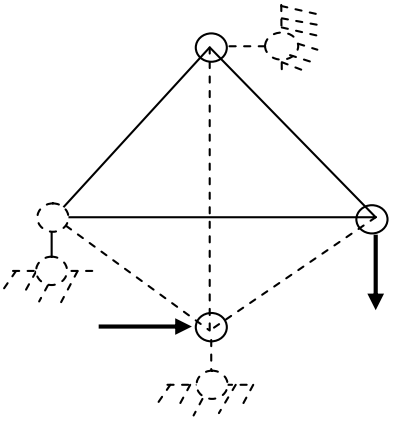


Рис. 26.1

1. Подготовим данные для расчета: назначим геометрические размеры системы и нагрузки, как показано на рис. 26.2.

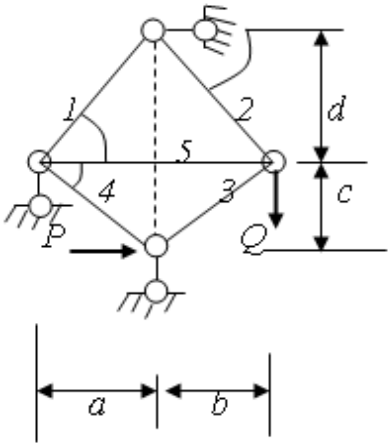


Рис. 26.2

Число искомых усилий $N = 8$: усилия в пяти стержнях и три опорные реакции.

Таким образом, исходные данные для расчета*: a, b ,c, d, P, Q, N.*

2. Вычислим длины стержней, а также синусы и косинусы углов наклонов наклонных стержней:

$$L1 = \sqrt{(a^2 + d^2)}; \quad L2 = \sqrt{(b^2 + d^2)};$$
$$L3 = \sqrt{(b^2 + c^2)}; \quad L4 = \sqrt{(a^2 + c^2)}. \qquad (1)$$

$$\sin\alpha_1 = d/L1, \ \cos\alpha_1 = a/L1,$$
$$\sin\alpha_2 = d/L2, \cos\alpha_2 = b/L2, \ \sin\alpha_3 = c/L3, \ \cos\alpha_3 = b/L3, \qquad (2)$$
$$\sin\alpha_4 = c/L4, \ \cos\alpha_4 = a/L4.$$

3. Обозначим номера узлов, затем мысленно вырежем узлы, а стержни и опорные связи заменим искомыми силами $x(1)$, $x(2)$, ..., $x(8)$ как показано на рис. 26.3.

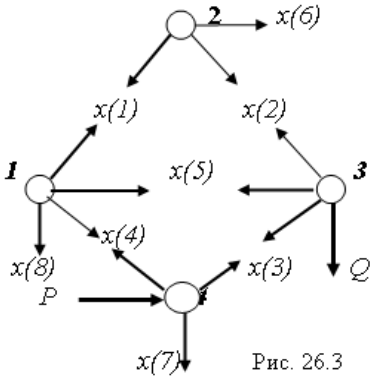


Рис. 26.3

4. Силы, действующие на каждый узел, проектируем на оси *X, Y*.

Узел 1: 1) $\Sigma X_{\text{УЗЛА 1}} = x(1)\cdot\cos\alpha_1 + x(4)\cdot\cos\alpha_4 + x(5) + 0 = 0;$

2) $\Sigma Y_{\text{УЗЛА 1}} = x(1)\cdot\sin\alpha_1 - x(4)\cdot\sin\alpha_4 - x(8) + 0 = 0.$

Узел 2: 3) $\Sigma X_{\text{УЗЛА 2}} = -x(1)\cdot\cos\alpha_1 + x(2)\cdot\cos\alpha_2 + x(6) + 0 = 0;$

4) $\Sigma Y_{\text{УЗЛА 2}} = -x(1)\cdot\sin\alpha_1 - x(2)\cdot\sin\alpha_2 + 0 = 0$.

Узел 3: 5) $\Sigma X_{\text{УЗЛА 3}} = -x(2)\cdot\cos\alpha_2 - x(3)\cdot\cos\alpha_3 - x(5) + 0 = 0$;

6) $\Sigma Y_{\text{УЗЛА 3}} = x(2)\cdot\sin\alpha_2 - x(3)\cdot\sin\alpha_3 - Q = 0$.

Узел 4: 7) $\Sigma X_{\text{УЗЛА 4}} = x(3)\cdot\cos\alpha_3 - x(4)\cdot\cos\alpha_4 + P = 0$;

8) $\Sigma Y_{\text{УЗЛА 4}} = x(3)\cdot\sin\alpha_3 + x(4)\cdot\sin\alpha_4 - x(7) + 0 = 0$.

5. Запишем эту систему в виде таблицы. Коэффициенты при неизвестных запишем в более краткой форме по номерам стержней:

Sn1, Sn2 – синусы; аналогично косинусы.

6. Сформируем матрицу коэффициентов и свободных членов, используя таблицу.

Таблица 26.1

№ узла	*X*1	*X*2	*X*3	*X*4	*X*5	*X*6	*X*7	*X*8	*B*		Номера Строк
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
1	Cs1 Sn1			Cs4 –Sn4	1			 –1	+0 +0	=0 =0	1 2

Окончание табл. 26.1

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
2	–Cs1 –Sn1	Cs2 –Sn2				1			+0 +0	=0 =0	3 4
3		–Cs2 Sn2	–Cs3 –Sn3		–1				+0 –Q	=0 =0	5 6
4			Cs3 Sn3	–Cs4 Sn4			 –1		+P +0	=0 =0	7 8

$A(1,1) = \cos1$; $A(1,4) = \cos4$; $A(1,5) = +1$;
$A(2,1) = \sin1$; $A(2,4) = -\sin4$; $A(2,8) = -1$;
$A(3,1) = -\cos1$; $A(3,2) = \cos2$; $A(3,6) = +1$;
$A(4,1) = -\sin1$; $A(4,2) = -\sin2$;
$A(5,2) = -\cos2$; $A(5,3) = -\cos3$; $A(5,5) = -1$;
$A(6,2) = \sin2$; $A(6,3) = -\sin3$; $B(6) = -Q$;
$A(7,3) = \cos3$; $A(7,4) = -\cos4$; $B(7) = +P$; (3)

$A(8,3) = \sin 3$; $A(8,4) = \sin 4$; $A(8,7) = -1$.

Остальные коэффициенты при неизвестных и свободные члены равны нулю.

7. Схема программы расчета фермы.

Имя программы *FERMA8* (по размерности решаемой системы уравнений).

Массивы $A(10\times10)$, $B(10)$, $X(10)$.

1. Задание исходных данных оператором *DATA* и вывод их на экран.

2. Вычисление геометрических характеристик наклонных стержней по формулам (1), (2).

3. Формирование матрицы коэффициентов и столбца свободных членов по формулам (3).

4. Решение системы линейных уравнений методом Гаусса с выбором ведущего элемента по матрице по алгоритму, изложенному в п. 23.3.

Ниже приведен текст программы на Фортране, в которой метки совпадают с номерами пунктов схемы программы.

```
program FERMA8
Real A(15,15),B(15),C(15,15),D(15),XX(15),X(15)
Real  Y(15),Q,L1, L2, L3, L4
INTEGER r,s,Ni(15),Nj(15)
PRINT *,'FERMA8 S RESHENIJEM OBSHCHEJ SISTEMI URAVNENIJ'

1 DATA AA,BB,CC,DD,P,Q/4.,4.,3.,3.,6.,12./
      PRINT *,'AA,BB,CC,DD=',AA,BB,CC,DD
      PRINT *,'P,Q=',P,Q
2  L1=SQRT(AA**2+DD**2)
      L2=SQRT(BB**2+DD**2)                  ! Вычисление
      L3=SQRT(BB**2+CC**2)                  ! геометрических
      L4=SQRT(AA**2+CC**2)                  ! параметров
      PRINT *,'L1, L2, L3, L4=',L1, L2, L3, L4    ! стержней

      Sn1=dD/L1;  Cs1=aA/L1;
```

```
      Sn2=dD/L2;  Cs2=bB/L2
      Sn3=cC/L3;  Cs3=bB/L3
      Sn4=cC/L4;  Cs4=aA/L4
      PRINT *,'SN1,SN2,SN3,SN4=',SN1,SN2,SN3,SN4
      PRINT *,'CS1,CS2,CS3,CS4=',CS1,CS2,CS3,CS4
                  ! Формирование матрицы A и столбца B

6  A(1,1) = Cs1;      A(1,4) = Cs4;   A(1,5) = +1.
      A(2,1) = Sn1;   A(2,4) = -Sn4;  A(2,8) = -1.
      A(3,1) = -Cs1;  A(3,2) = Cs2;   A(3,6) = +1.
      A(4,1) = -Sn1;  A(4,2) = -Sn2;
      A(5,2) = -Cs2;  A(5,3) = -Cs3;  A(5,5) = -1.
      A(6,2) = Sn2;   A(6,3) = -Sn3;          B(6) = -Q
      A(7,3) = Cs3;   A(7,4) = -Cs4;          B(7) = P
      A(8,3) = Sn3;   A(8,4) = Sn4;   A(8,7) = -1.
  N=8
20 do i=1,n; D(i)=b(i);  do j=1,n        Решение системы линейных
      C(i,j)=A(i,j);  end do                 уравнений методом Гаусса
  end do                                 с выбором ведущего элемента
      Pause '30 formirovanije Ni(n),Nj(n) i vivod ih'      по матрице
        k=0
30 DO I=1,N;k=k+1; Ni(i)=k;   Nj(i)=k;   End do;       PAUSE '30'
      PRINT 32,'31.Ni=',(Ni(i),i=1,n)          Формирование Ni, Nj
      PRINT 32,'Nj=',(Nj(j),j=1,n)     ;
32  Format(1x,A,15i5)
40 PRINT *,'40.  Ishodnaja tablica:'          Вывод таблицы A,B,
CaII VIVOD2(n,A,B,Ni,Nj)                      Ni, Nj
50    Pause '50 priamoi hod Gaussa
DO 54 K=1,N-1                                 K – ведущий элемент
                    PRINT *,'S H A G  ',k
      CALL POISK1(N,K,A,Q,r,s)       Поиск наибольшего элемента
      PRINT *,'Q,k,r,s=',Q,k,r,s,' ZAMTNA STROK K,r'; Pause
51 CaII ZAMENAr(N,K,r,A,B,Ni)                 Замена строк
                                  CaII VIVOD2(n,A,B,Ni,Nj)
                              PAUSE 'ZAMENA STOLBCOV k,s'
52       CALL ZAMENAs(N,k,s,A,NJ)             Замена столбцов
                                  CaII VIVOD2(n,A,B,Ni,Nj)
```

```
                                PAUSE 'SHAG PRIAMOGO HODA GAUSSA'
  DO i=k+1,N                                  Прямой ход метода Гаусса
     B(i)=B(i)-B(k)/A(k,k)*A(i,k)
     DO j=n,k,-1
       A(i,j)=A(i,j)-A(k,j)/A(k,k)*A(i,k); end do
  end do;
54 continue
PRINT *,'R E Z U L T A T   PRIAMOGO HODA:';  pause
                                        Call VIVOD2(n,A,B,Ni,Nj)
pause '60 OBRATNIJ HOD GAUSSA'                Обратный ход
                                                метода Гаусса
60 xx(n)=B(n)/A(n,n)
do i=n-1,1,-1
  sum=0;  do j=i+1,n; SUM=SUM+A(i,j)*xx(j); end do
  XX(i)=(B(i)-SUM)/A(i,i)
END DO
PRINT *,'XX=',(XX(j),j=1,n);
PRINT *,'NJ=',(NJ(j),j=1,n)
70 PAUSE 'UPORIADOCHENIJE XX'
DO i=1,N;X(NJ(i))=XX(i); END DO       Упорядочение вектора XX
PRINT *,'X=',(X(j),j=1,n);                  Вывод результата
PRINT *,'j=',(j,j=1,N)
pause '80 proverka:y(i)=SUM(C(i,j)*X(j)-D(i)'   Проверка решения
DO i=1,N;
      SUM=0; do j=1,n; SUM=SUM+C(i,j)*X(j);end do
 Y(i)=SUM-D(i)
end do
                                PRINT *,'Y=',(Y(i),i=1,n)
Stop
End Program FERMA8
Subroutine VVOD1(n,A,B)          Подпрограмма ввода таблицы
Real A(15,15),B(15)              A(m×n), B(m) из внешнего файла
READ(1,*)n
do i=1,n;    READ(1,*)(A(i,j),j=1,n),B(i);   end do
End
Subroutine VIVOD1(n,A,B,v)                Подпрограмма вывода
Real A(15,15),B(15)                       матрицы A, B
```

```
Character v*5
Print *,'Sistema uravnenij',v
do i=1,n;     write(*,10)(A(i,j),j=1,n),B(I);    end do
10 format(1x,15f6.2)
end
```

```
Subroutine VIVOD2(n,A,B,Ni,Nj)
Real A(15,15),B(15)
INTEGER Ni(10),NJ(10)
      Print *,'V i v o d 2'; pause
WRITE(*,11)(NJ(j),j=1,N)
11 FORMAT(1x,4x, 15i5)
do i=1,n;     write(*,21)Ni(i),(A(i,j),j=1,n),B(i);
21 Format(1x,i3,1x,15f5.1)
end do
end
```

Подпрограмма вывода таблицы *A, B, N, Ni, Nj* на экран по формату

```
SUBROUTINE POISK1(N,K,A,Q,r,s)
REAL A(15,15)
INTEGER r,s,i,j,k
Q=0.0; r=k; s=k
Do i=k,n; Do j=k,N
      IF(ABS(A(i,j)).GT.ABS(Q))then
      Q=A(i,j); r=i;   s=j
       end if
END do;  END do
End
```

Подпрограмма поиска наибольшего по модулю $A(i,j),\ i = k, \ldots, N;\ j = k, \ldots, N$

```
SUBROUTINE ZAMENAr(N,K,r,A,B,Ni)
Real A(15,15),B(15)
Integer N,K,r,Ni(15),j
  L=Ni(k); M=Ni(r);  Ni(k)=M;  Ni(r)=L  !
  Do j=1,N;  u=A(k,j);  v=A(r,j)
                    A(k,j)=v;          A(r,j)=u; End do
u=B(k);  v=B(r);  B(k)=v;  B(r)=u
End
```

Подпрограмма замены строк *k*, *r*

```
SUBROUTINE ZAMENAs(N,k,s,A,NJ)
REAL A(15,15)
```

Подпрограмма замены столбцов *k,s*

```
INTEGER N,k,s,NJ(15),i
      L=NJ(k); M=NJ(s); NJ(k)=M; NJ(s)=L
DO i=1,n
u=A(i,k);v=A(i,s);A(i,k)=v;A(i,s)=u
End do
 End SUBROUTINE ZAMENAs
```

Результаты расчета

```
FERMA8 S RESHENIJEM OBSHCHEJ SISTEMI URAVNENIJ
AA,BB,CC,DD=   4.000000      4.000000      3.000000      3.000000
P,Q=      6.000000      12.000000
L1, L2, L3, L4=   5.000000      5.000000      5.000000      5.000000
SN1,SN2,SN3,SN4=   6.000000E-01   6.000000E-01   6.000000E-01
6.000000E-01
CS1,CS2,CS3,CS4=   8.000000E-01   8.000000E-01   8.000000E-01
8.000000E-01
30 formirovanije Ni(n),Nj(n) i vivod ih
31.Ni=   1   2   3   4   5   6   7   8
Nj=   1   2   3   4   5   6   7   8
40.  Ishodnaja tablica:
       1    2     3     4     5     6     7     8
  1   .8   .0    .0    .8   1.0    .0    .0    .0    .0
  2   .6   .0    .0   -.6    .0    .0    .0  -1.0    .0
  3  -.8   .8    .0    .0    .0   1.0    .0    .0    .0
  4  -.6  -.6    .0    .0    .0    .0    .0    .0    .0
  5   .0  -.8   -.8    .0  -1.0    .0    .0    .0    .0
  6   .0   .6   -.6    .0    .0    .0    .0    .0  -12.0
  7   .0   .0    .8   -.8    .0    .0    .0    .0   6.0
  8   .0   .0    .6    .6    .0    .0  -1.0    .0    .0

50 priamoi hod Gaussa
R E Z U L T A T   PRIAMOGO HODA:  ! прямой ход
       5     8     6     7     1     2     3     4
  1  1.0    .0    .0    .0    .8    .0    .0    .8    .0
  2   .0  -1.0    .0    .0    .6    .0    .0   -.6    .0
  3   .0   .0   1.0    .0   -.8    .8    .0    .0    .0
```

```
 8   .0   .0   .0 -1.0   .0   .0   .6   .6   .0
 5   .0   .0   .0   .0   .8  -.8  -.8   .8   .0
 4   .0   .0   .0   .0   .0 -1.2  -.6   .6   .0
 6   .0   .0   .0   .0   .0   .0  -.9   .3-12.0
 7   .0   .0   .0   .0   .0   .0   .0  -.5 -4.7

60 OBRATNIJ HOD GAUSSA            ! обратный ход
XX=     10.000000       2.999999      -6.000001     -15.000000
        -3.750001       3.750000     -16.250000      -8.749999
NJ=        5                8             6              7
           1                2             3              4

UPORIADOCHENIJE XX                ! окончательный результат
X=      -3.750001       3.750000     -16.250000      -8.749999
        10.000000      -6.000001     -15.000000       2.999999
j=         1                2             3              4
           5                6             7              8
80 proverka:y(i)=SUM(C(i,j)*X(j)-D(i)   ! проверка
Y=  0.000000E+00  0.000000E+00  0.000000E+00  4.827976E-07
  0.000000E+00      24.000000     -12.000000   0.000000E+00
```

Проверим теперь равновесие узла 4:

$$\Sigma X_{\text{УЗЛА }4} = x(3)\cdot\cos\alpha_3 - x(4)\cdot\cos\alpha_4 + P = 0;$$

$$\Sigma Y_{\text{УЗЛА }4} = x(3)\cdot\sin\alpha_3 + x(4)\cdot\sin\alpha_4 - x(7) + 0 = 0;$$

$$(-16{,}25)\cdot 0{,}8 - (-8{,}75\cdot 0{,}8) + 6 = 0;$$

$$(-16{,}25)\cdot 0{,}6 + (-8{,}75\cdot 0{,}6) - (-15) = 0.$$

Задача 26.2. Использование подпрограммы решения системы двух уравнений для расчета фермы

Решим задачу 26.1 другим способом. Для удобства читателя повторим здесь рис. 26.2. Несколько изменив рис. 26.3, получим рис. 26.4.

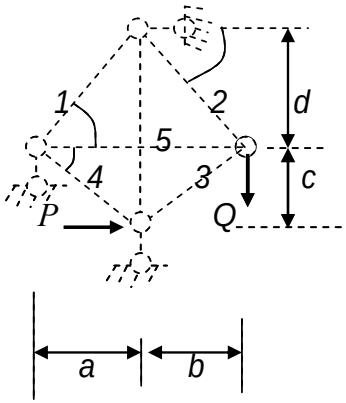


Рис. 26.2

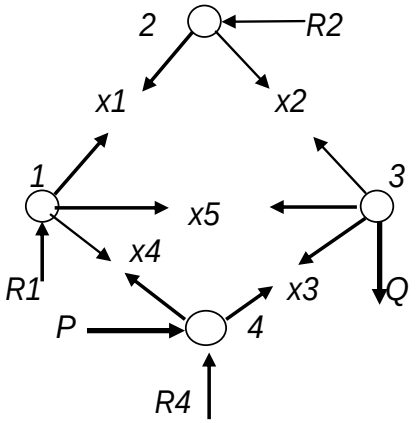


Рис. 26.4

1. Подготовим данные для расчета: назначим геометрические размеры системы и нагрузки, как показано на рис. 26.2.

Число искомых усилий N=8: усилия в пяти стержнях и три опорные реакции.

Таким образом, исходные данные для расчета: *a, b, c, d, P, Q.*

2. Вычислим длины стержней, а также синусы и косинусы углов наклонов наклонных стержней:

$$L1 = \sqrt{(a^2 + d^2)}; \quad L2 = \sqrt{(b^2 + d^2)};$$
$$L3 = \sqrt{(b^2 + c^2)}; \quad L4 = \sqrt{(a^2 + c^2)}. \qquad (1)$$

$$\sin\alpha_1 = d/L1, \ \cos\alpha_1 = a/L1,$$
$$\sin\alpha_2 = d/L2, \ \cos\alpha_2 = b/L2, \ \sin\alpha_3 = c/L3, \ \cos\alpha_3 = b/L3, \qquad (2)$$
$$\sin\alpha_4 = c/L4, \ \cos\alpha_4 = a/L4.$$

Для последующих вычислений рис. 26.2 дополним и получим рис. 26.5. Тогда $U = a+b$, $h = c+d$.

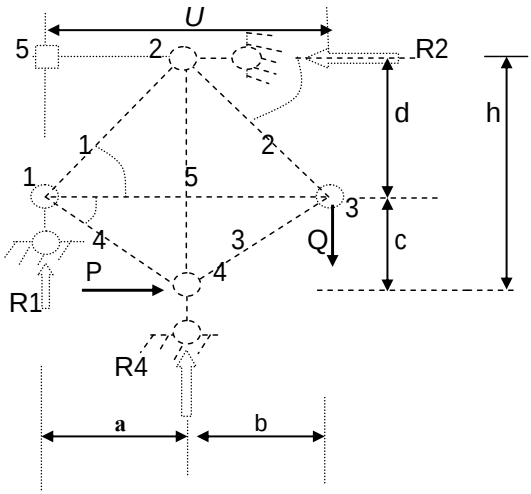


Рис. 26.5. Расчетная схема фермы с опорными реакциями
и дополнительной моментной точкой 5

3. Вычислим опорные реакции:

$\sum M_2 = -R1 \cdot a + P \cdot h - Q \cdot b = 0$, отсюда $R1 = (P \cdot h - Q \cdot b) / a.$

$\sum M_5 = +R4 \cdot A + P \cdot H - Q \cdot U = 0$, отсюда $R4 = (-P \cdot H + Q \cdot U) / A.$

$\sum X = -R2 + P = 0$, отсюда $R2 = P.$

Контроль: $\sum M_3 = SM3 = -R1 \cdot U + R2 \cdot H + P \cdot C = 0$?

4. Обозначим номера узлов, затем мысленно вырежем узлы, а стержни и опорные связи заменим искомыми силами $x(1)$, $x(2)$, ..., $x(8)$, как показано на рис. 26.4.

5. Подготовим подпрограмму решения системы двух уравнений:

```
SUBROUTINE ALGUR2(a1,b1,c1,a2,b2,c2,X,Y)
D= a1*b2–a2*b1;   D1 = c1*b2–c2*b1;      D2= a1*c2–a2*c1
```

x = – d1/d; y = – d2/d; *END SUBROUTINE ALGUR2*

6. Силы, действующие на каждый узел, проектируем на оси *X, Y.*

Для краткости **упростим написание** синусов и косинусов, чтобы они совпадали с обозначениями простых переменных в программе: Sn1, Cs1, Sn2, Cs2, .., а знак умножения заменим звездочкой. Будем одновременно записывать фрагменты программы. Фрагменты будущей программы и свободные члены уравнений выделим **жирным шрифтом**. В число свободных членов уравнений входят **уже вычисленные усилия**. Свободные члены подчеркнуты пунктирными линиями.

Узел 2: в нем два неизвестных усилия *x*1, *x*2.

$\Sigma X_{\text{УЗЛА 2}} = -x1\times Cs1 + x2\times Cs2 - R2 = 0;$

$\Sigma Y_{\text{УЗЛА 2}} = -x1\times Sn1 - x2\times Sn2 + 0 = 0;$

CALL ALGUR2(–CS1, +CS2, –R2, –SN1, –SN2, 0. X1,X2).

Узел 1: в нем неизвестны *x*4, *x*5.

$\Sigma X_{\text{УЗЛА 1}} = +x4\times Cs4 + x5\times 1. + x1\times Cs1 = 0;$

$\Sigma Y_{\text{УЗЛА 1}} = -x4\times Sn4 + x5\times 0 + R1 + x1\times Sn1 = 0;$

*CALL ALGUR2(+CS4, +,1.,x1*CS1, –SN4, 0, R1+X1*SN1,. X4,X5)*

Узел 3: в нем неизвестно только усилие *x*3.

$\Sigma Y_{\text{УЗЛА 3}} = x2\times Sn2 - x3\times Sn3 - Q = 0, \quad x3 = (x2\times Sn2 - Q)/Sn3.$

Контроль:

Узел 4: 7) $\Sigma X_{\text{УЗЛА 4}} = SX4 = x3\times Cs\alpha_3 - x4\times Cs\alpha_4 + P = 0;$

8) $\Sigma Y_{\text{УЗЛА 4}} = SY4 = x3\times Sn3 + x4\times Sn4 + r4 = 0?$

7. Текст программы:

```
real L1,L2,L3,L4
DATA  a,b,c,d,P,Q,/4.,4.,3.,3.,6.,12./
      PRINT *,'A,B,C,D=',A,B,C,D
      PRINT *,'P,Q=',P,Q
    2  L1=SQRT(A**2+D**2);              L2=SQRT(B**2+D**2)
      L3=SQRT(B**2+C**2);        L4=SQRT(A**2+C**2)
      PRINT *,'L1, L2, L3, L4=',L1, L2, L3, L4
```

```
      Sn1=dD/L1;  Cs1=aA/L1;        Sn2=dD/L2;  Cs2=bB/L2
      Sn3=cC/L3;  Cs3=bB/L3;        Sn4=cC/L4;  Cs4=aA/L4
      PRINT *,'SN1,SN2,SN3,SN4=',SN1,SN2,SN3,SN4
      PRINT *,'CS1,CS2,CS3,CS4=',CS1,CS2,CS3,CS4
  3  h = c + d;   U = A + B
     R1 = (P*h – Q*b)/a
     R4 = (– P*H + Q*U)/A
    R2 = P;              SM3 =  – R1*U + R2*H + P*C
  CAL ALGUR2(–CS1,CS2,–R2,  –SN1,–SN2,+0,  X1, X2)
              PRINT*, 'X1,X2=', X1,X2
  CALL ALGUR2(+CS4, +,1.,x1*CS1,  –SN4, 0,  R1+X1*SN1,. X4,X5)
              print *,' X4,X5=', X4,X5
  x3 = (x2* Sn2  - Q)/Sn3;   print *,'x3=',x3
   SX4 =  x3* Csα3 - x4*Csα4  + P
   SY4 = x3* Sn3 + x4* Sn4 – x7) + r4;        print *,sx4,sy4=', sx4,sy4
  END
  SUBROUTINE ALGUR2(a1,b1,c1,a2,b2,c2,X,Y)
  D= a1*b2–a2*b1;   D1 = c1*b2–c2*b1;     D2= a1*c2–a2*c1
  x = – d1/d;  y = – d2/d;                 END SUBROUTINE ALGUR2
```

Результаты вычислений

```
  A,B,C,D=      4.000000      4.000000      3.000000      3.000000
  P,Q=      6.000000      12.000000
  L1, L2, L3, L4=      5.000000      5.000000      5.000000      5.000000
  SN1,SN2,SN3,SN4=  6.000000E-01   6.000000E-0       6.000000E-0
6.000000E-01
  CS1,CS2,CS3,CS4=  8.000000E-01   8.000000E-01     8.000000E-01
8.000000E-01
  X1,X2=   -3.750000       3.750000
  X4,X5=     -8.750000       10.000000
  x3=     -16.250000
  sx4,sy4=  -8.940697E-08   -5.960464E-07
  Press any key to continue
```

27. Организация циклов в задачах механики балки под нагрузкой

От студента не требуется научиться решать приведенные ниже задачи, решения их даны. Если задача сложна для понимания, следует обратить внимание на полученные формулы и алгоритмы. Необходимо уметь их программировать. Понимание сути задач облегчит работу.

Задача 27.1. Однопролетная балка, одна сила

Дано: P, L, x.

Вычислить: R_A, R_B, момент внешних сил под неподвижной силой P (рис. 27.1).

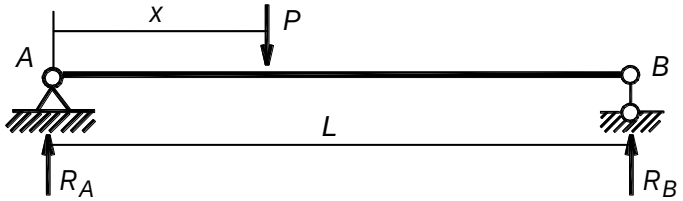
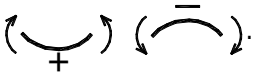


Рис. 27.1. Однопролетная балка, одна сила

$$\sum M_A = -R_B \cdot L + P \cdot x = 0 \Rightarrow R_B = P \cdot x / L,$$

$$\sum M_B = R_A L - P(L - x) = 0 \Rightarrow R_A = P(L - x)/x.$$

Контроль: $\sum Y = SY = R_A + R_B - P = 0$?

Правило знаков моментов внешних сил:  .

Моменты внешних сил слева: $ML = R_A \cdot x$.

Моменты внешних сил справа: $MP = R_B(L - x)$.

Задача 27.2. Однопролетная балка, одна сила, n сечений

Дано: P, L, n,

где n – число расчетных сечений, которые расположены с одинаковым шагом (рис. 27.2).

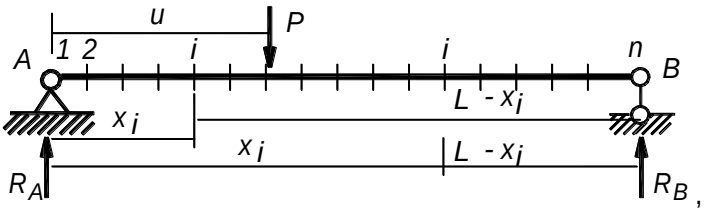


Рис. 27.2. Однопролетная балка, одна сила, n сечений

Вычислить: R_A, R_B, а также ML_i – момент внешних сил слева в расчетных сечениях; MP_i – то же, справа.

1. $dx = L/(n-1)$, $R_B = P \cdot u / L$, $R_A = P(L-u)/L$.
2. $i = 1, \ldots, n$; $x_i = dx(i-1)$.

$x_i \le u$? $\xrightarrow{нет}$ $ML_i = R_A x_i - P(x_i - u)$; $MP_i = R_B(L - x_i)$;

$\downarrow да$

$ML_i = R_A x_i$;

$MP_i = R_B(L - x_i) - P(u - x_i)$.

Следующее i.

Задача 27.3. Однопролетная балка, подвижная сила, одно сечение

Дано: P, L, n, x_k;

где P – подвижная сила;

k – расчетное сечение (рис. 27.3).

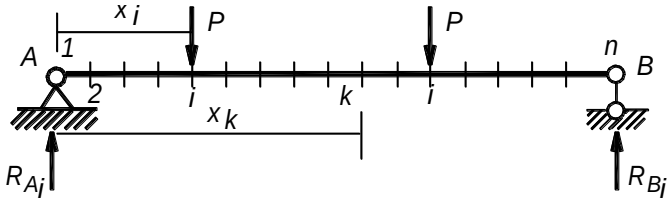


Рис. 27.3. Однопролетная балка, подвижная сила, одно сечение

Требуется вычислить момент внешних сил в сечении k при всех положениях нагрузки. Алгоритм предусматривает учет двух вариантов: 1) сила P расположена в произвольном сечении i слева

от расчетного сечения k; 2) сила P расположена в произвольном сечении i справа от расчетного сечения k.

1. $dx = L/(n-1)$.
2. $i = 1, \ldots, n$.
3. $x_i = dx\,(i-1)$.
4. $R_{Ai} = P(L - x_i)/L$.
5. $R_{Bi} = P \cdot x_i / L$.
6. $x_i \le x_k$? $\xrightarrow{да}$ 7. $Mk_i = R_{Bi}(L - x_k)$.

↓ *нет*

8. $Mk_i = R_A \cdot x_k$.
9. Следующее i.

Вывести i, R_{Ai}, R_{Bi}, M_{ki} по формату для i = 1,2,..., n.

Задача 27.4. Однопролетная балка, вектор сил, найти опорные реакции

Дано: L, n, $\overline{P}(n)$.

Вычислить: 1) R_A, R_B – опорные реакции;

2) ML_i $(i = 1, \ldots, n)$ – моменты внешних сил слева;

3) MP_i $(i = 1, \ldots, n)$ – моменты внешних сил справа.

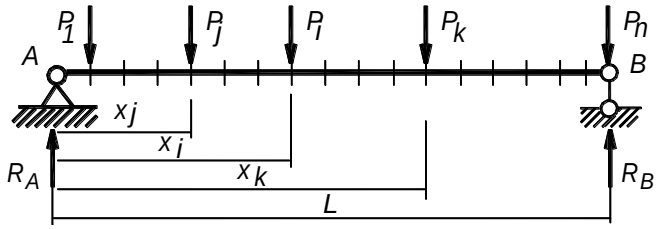


Рис. 27.4. Однопролетная балка, вектор сил

Расчетные формулы

1) $\sum M_A = -R_B \cdot L + \sum_{i=1}^{n} P_i \cdot x_i = 0 \quad \Rightarrow \quad R_B = (\sum_{i=1}^{n} P_i\, x_i)/L$;

$\sum M_B = R_A \cdot L - \sum_{i=1}^{n} P_i\,(L - x_i) = 0 \quad \Rightarrow \quad R_A = (\sum_{i=1}^{n} P_i\,(L - x_i))/L$;

2) для $i = 1, \ldots, n$:

$$ML_i = R_A \cdot x_i - \sum_{j=1}^{i} P_j \, (x_i - x_j) \, ;$$

$$MP_i = R_B \, (L - x_i) - \sum_{k=i+1}^{n} P_k \, (x_k - x_i) \, .$$

Вывести i, x_i, ML_i, MP_i по формату.

28. Программирование вычисления реакций в трехшарнирной раме

На рис. 28.1 показана трехшарнирная рама с опорами на разных уровнях.

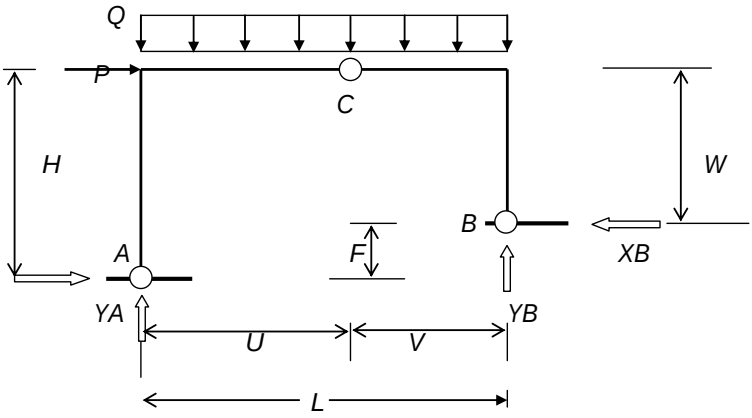


Рис. 28.1. Определение реакций в трехшарнирной раме с опорами на разных уровнях

Схема программы

1. Ввод данных: *U, V, W, H, P, Q.*
2. *L = U + V;* *F = H – W.*
3. Подготовим подпрограмму решения системы двух уравнений.

```
SUBROUTINE ALGUR2(a1,b1,c1,a2,b2,c2,X,Y)
D= a1*b2–a2*b1;   D1 = c1*b2–c2*b1;     D2= a1*c2–a2*c1
x = – d1/d;  y = – d2/d;                END SUBROUTINE ALGUR2
```

4. Приравняем к нулю сумму моментов всех сил относительно левой опоры и сумму моментов сил, приложенных к правой части рамы, относительно шарнира *C*. Полученную систему уравнений решим с помощью подпрограммы *ALGUR*2 и в результате найдем опорные реакции *XB,YB.*

$$\sum M_A = +XB*F + YB*L - P*H - Q*L*L/2. = 0;$$

$$\sum M_C^{\text{ПРАВОЙ ЧАСТИ}} = -XB*W + YB*V - Q*V*V/2 = 0;$$

*CALL ALGUR2(F, L, – P*H – Q*L*L/2, –W, +V, – Q*V*V/2., XB,YB).*

5. Аналогично: $\sum M_B = + XA*F - YA*L - P*W + Q*L*L/2. = 0;$

$$\sum M_C^{\text{ЛЕВ. ЧАСТИ}} = +XA*H - YA*U + Q*U*U/2. = 0.$$

Подчеркнуты пунктирными линиями свободные члены в уравнениях.

*CALL ALGUR2(F, –L, – P*W + Q*L*L/2, +H, –U, +*
*+ Q*U*U/2., XA, YA).*

6. Контроль: $\sum X = SX = XA - XB + P = 0.$

$$\sum Y = SY = YA + YB - Q \cdot L = 0?$$

7. Текст программы:

```
PROGRAM RAMAT
REAL L
U = 6;      V = 4;  H = 6;  W = 4;  P = 10;Q = 6
PRINT *,'U,V,H,W=',U,V,H,W
PRINT *,'P,Q=',P,Q
L = U + V;  F = H - W
PRINT *,'L,F=',L,F
CALL ALGUR2(F,L,-P*H-Q*L*L/2,-W,+V,-Q*V*V/2.,XB,YB)
PRINT *,'XB,YB=',XB,YB
CALL ALGUR2(F, -L,  - P*W + Q*L*L/2,   +H, -U, + Q*U*U/2.,
XA, YA)
```

```
PRINT *,'XA,YA=',XA,YA
SX = XA - XB + P                ! проверка
SY =  YA +YB - Q*L
PRINT *,'SX,SY=',SX,SY
STOP

END
SUBROUTINE ALGUR2(a1,b1,c1,a2,b2,c2,X,Y)
D= a1*b2-a2*b1;    D1 = c1*b2-c2*b1;      D2= a1*c2-a2*c1
x = - d1/d;  y = - d2/d;                  END SUBROUTINE ALGUR2
```

Результаты вычислений по программе *RAMAT*

```
XB,YB=      20.000000       32.000000
XA,YA=      10.000000       28.000000
SX,SY=   0.000000E+00    0.000000E+00
```

Оглавление

Учебное издание

СОЛОДОВ Борис Павлович

СБОРНИК ЗАДАЧ С РЕШЕНИЯМИ ПО ПРОГРАММИРОВАНИЮ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ Fortran PowerStation

Методическое пособие по дисциплине «Информатика»
для студентов вузов специальности
1-70 02 01 «Промышленное и гражданское строительство»
заочной формы обучения

Редактор Т.А. Подолякова
Компьютерная верстка Д.К. Измайлович

Подписано в печать 16.12.2010.
Формат 60×84 1/16. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 7,27. Уч.-изд. л. 5,68. Тираж 300. Заказ 2.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.