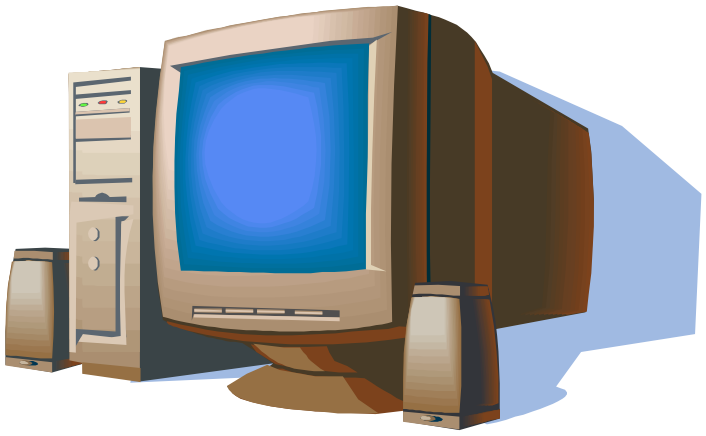
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Ефименко К.Н., Добровольский Ю.Н., Ильяшенко В.С.

МЕТОДИЧЕСКОЕ ПОСОБИЕ

«ОСНОВЫ АЛГОРИТМИЗАЦИИ»



Донецк 2006

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Ефименко К.Н., Добровольский Ю.Н., Ильяшенко В.С.

МЕТОДИЧЕСКОЕ ПОСОБИЕ
«ОСНОВЫ АЛГОРИТМИЗАЦИИ»

Утверждено
на заседании кафедры ВМиП
Протокол № 1 от 30 августа 2005 г.

Утверждено
методической комиссией ДонНТУ
Протокол № 4 от 27 октября 2005 г.

Донецк 2006

УДК 681.3.06 (071)

Методическое пособие. «ОСНОВЫ АЛГОРИТМИЗАЦИИ» / К.Н. Ефименко, Ю.Н. Добровольский, В.С. Ильяшенко – Донецк: ДонНТУ, 2006. – 75 с.

Методическое пособие содержит теоретический материал по основам алгоритмизации и способам составления блок-схем алгоритмов типовых структур. Приведены задания для контрольных работ и примеры их выполнения.

Предназначены для студентов заочной формы обучения всех специальностей.

Авторы: **К.Н. Ефименко**
Ю.Н. Добровольский
В.С. Ильяшенко

Отв. за выпуск: В.Н. Павлыш, доц.

СОДЕРЖАНИЕ

1. АЛГОРИТМЫ И СПОСОБЫ ИХ ОПИСАНИЯ

Алгоритм – это строгая последовательность арифметических и логических действий, которая однозначно определяет процесс вычисления результата в зависимости от исходных данных. Слово “алгоритм” произошло от арабского слова “algoritmi”, возникшего из имени хорезмийского математика IX века аль-Хорезми.

Основными свойствами алгоритма являются:

результативность – алгоритм должен обеспечивать получение результата за конечное число шагов;

определенность – применение алгоритма к однотипным исходным данным должно приводить к одному и тому же результату, независимо от пользователя;

массовость – возможность применения алгоритма к целому классу однотипных задач, различающихся исходными данными.

Наиболее распространенными способами описания алгоритма являются:

словесно-формульный – алгоритм записывается в виде текста с формулами по пунктам, определяющими последовательность действий;

аналитическая запись – алгоритм записывается с использованием формул или специальных символов, например, стенографических;

графический в виде блок-схемы – алгоритм изображается специальными геометрическими фигурами (блоками), связанными направляющими стрелками.

Образцом словесно-формульного описания алгоритма может служить следующий пример:

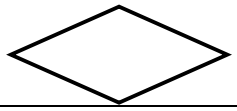
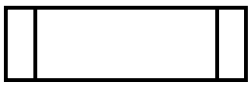
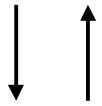
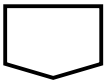
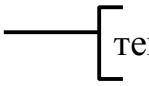
вычислить	1. *Ввести значения величин a и x.*
значение	2. *Выполнить сложение x и* 6 $\rightarrow$ x+6.
выражения	3. *Выполнить умножение a на* 2 $\rightarrow$ 2*a.*
y = 2 · a – (x + 6)	4. *Вычесть из полученного произведения значение полученной суммы* $\rightarrow$ 2*a-(x*+6*).*
	5. *Вывести значение полученного результата y.*

Программа, записанная на каком-либо языке программирования, также является формой представления алгоритма.

Однако наиболее удобным и наглядным способом представления алгоритма является графический в виде блок-схемы. При этом каждый логически завершенный этап вычислительного процесса изображается в виде специального геометри-

ческого символа – **блока**. Наиболее часто используемые графические символы представлены в таблице 1.

Таблица 1. Графические символы, применяемые при составлении блок-схем

№ п/п	Наименование блока	Обозначение	Выполняемое действие
1	Начало или конец алгоритма		Показывает начало или конец алгоритма
2	Ввод или вывод		Обеспечивает ввод или вывод данных в алгоритме
3	Арифметический		Выполняет арифметические вычисления
4	Логический		Выполняет проверку заданного логического условия
5	Модификации		Заголовок цикла «Для»
6	Предопределенный процесс		Вызов подпрограммы
7	Линии потока		Указывают связь и направление движения между блоками
8	Соединитель		Указывает связь между прерванными линиями потока
9	Межстраничный соединитель		Указывает связь между частями блок-схемы, которые расположены на разных страницах
10	Комментарии		Запись пояснения к блоку или к линии потока

При составлении блок-схемы алгоритма блоки записываются последовательно друг за другом и соединяются линиями потока информации, которые показывают направление движения по блок-схеме. Каждый блок алгоритма должен иметь вход и выход (исключение составляют блоки начала и конца алгоритма). При этом может быть несколько входящих в блок линий потока информации и только один выходящий поток (исключения составляют логический блок и блок модификации). Несколько линий потока могут объединяться в одну линию, но

одна линия потока информации не может разветвляться на несколько потоков. В блок-схеме любой путь движения из блока «Начало» алгоритма должен привести в блок «Конец» алгоритма.

В общем случае любой алгоритм может состоять из трех частей: ввод исходных данных, вычисление требуемых величин и вывод полученных результатов. При этом каждый блок в блок-схеме должен быть пронумерован.

Существует три основных типовых структуры алгоритма:

1. Линейный вычислительный процесс.
2. Разветвляющийся вычислительный процесс.
3. Циклический вычислительный процесс.

Любой алгоритм сложной структуры может быть получен путем комбинированного использования типовых структур.

2. АЛГОРИТМЫ ЛИНЕЙНОЙ СТРУКТУРЫ

В **линейном вычислительном процессе** все действия выполняются в строгой последовательности друг за другом. Таким образом, существует только один путь, по которому можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

Пример. Составить алгоритм вычисления площади треугольника с заданными сторонами ***a, b*** и ***c***.

Как известно из курса геометрии площадь треугольника, заданного длинами сторон, можно вычислить, используя формулу Герона. Разрабатываемый алгоритм (рис. 2.1) должен обеспечивать ввод исходных данных, т.е. значений длин сторон треугольника ***a, b*** и ***c*** (блок 2). Затем, используя введенные значения, вычислять полупериметр ***P*** (блок 3) и значение площади треугольника ***S*** (блок 4). После завершения вычислений необходимо

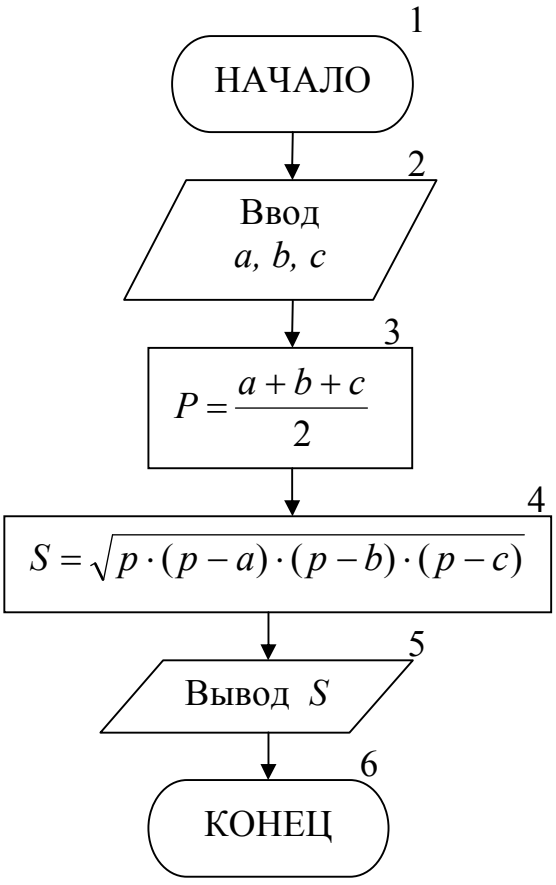


Рисунок 2.1. Алгоритм линейной структуры

вывести результат, т.е. полученное значение площади треугольника ***S*** (блок 5).

Однако решение абсолютного большинства инженерных задач невозможно свести к алгоритмам линейной структуры.

3. АЛГОРИТМЫ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Разветвляющийся вычислительный процесс позволяет выбрать один из нескольких вариантов решения поставленной задачи в зависимости от выполнения некоторых условий. Таким образом, существует несколько различных путей, по которым можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

Для реализации процесса выбора одного из двух вариантов решения используется логический блок (блок проверки условий), приведенный на рисунке 3.1. При входе в этот блок выполняется проверка логического условия (обычно математического неравенства). Если результат проверки условия «Истина», т.е. условие выполняется, то происходит переход к выполнению блоков, стоящих по ветви «+». В противном случае, т.е. когда проверяемое условие не выполняется, происходит переход к выполнению блоков, стоящих по ветви «–».

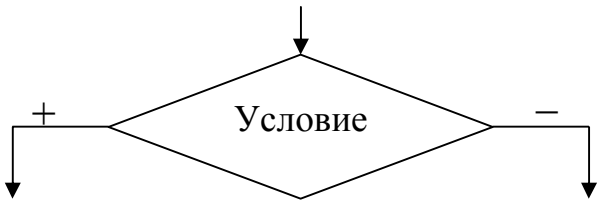


Рисунок 3.1. Логический блок

Если требуется выбрать один из трех и более вариантов решения, то необходимо использовать вложенные логические блоки. В случае, когда действия должны выполняться только по одной из ветвей логического блока, их необходимо реализовывать по ветви «+», подобрав соответствующее условие (рис. 3.2).

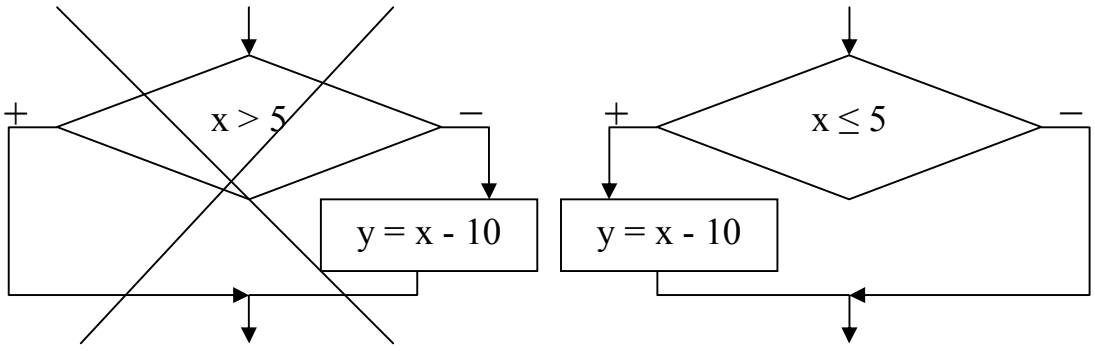


Рисунок 3.2. Рациональное использование логического блока

При выполнении вычислений необходимо учитывать область определения математических функций. Следовательно, вначале необходимо проверить воз-

можность вычисления данного математического выражения при текущих значениях исходных данных, т.е. проверить «**аномалию**». К наиболее часто встречающимся «аномалиям» относятся: операция деления (на 0 делить нельзя), вычисление квадратного корня (подкоренное выражение должно быть ≥ 0), вычисление логарифма (выражение под знаком логарифма должно быть > 0), вычисление tg, ctg. В случае возникновения «аномалии» (невозможно выполнить вычисления) необходимо пропустить все действия, которые зависят от вычисляемой величины, и перейти в ту часть алгоритма, где можно продолжить вычисления.

Пример. Составить блок-схему алгоритма, вычисляющего значение ***y*** по одной из трех формул, в зависимости от значения ***x***. Блок-схема алгоритма приведена на рисунке 3.3.

$$y=\begin{cases} x^2-4x, & \textit{если} \quad x\ge 4 \\ \sqrt{x-1.5}, & \textit{если} \quad 2\le x<4 \\ \dfrac{3x}{x+1}, & \textit{если} \quad x<2 \end{cases}$$

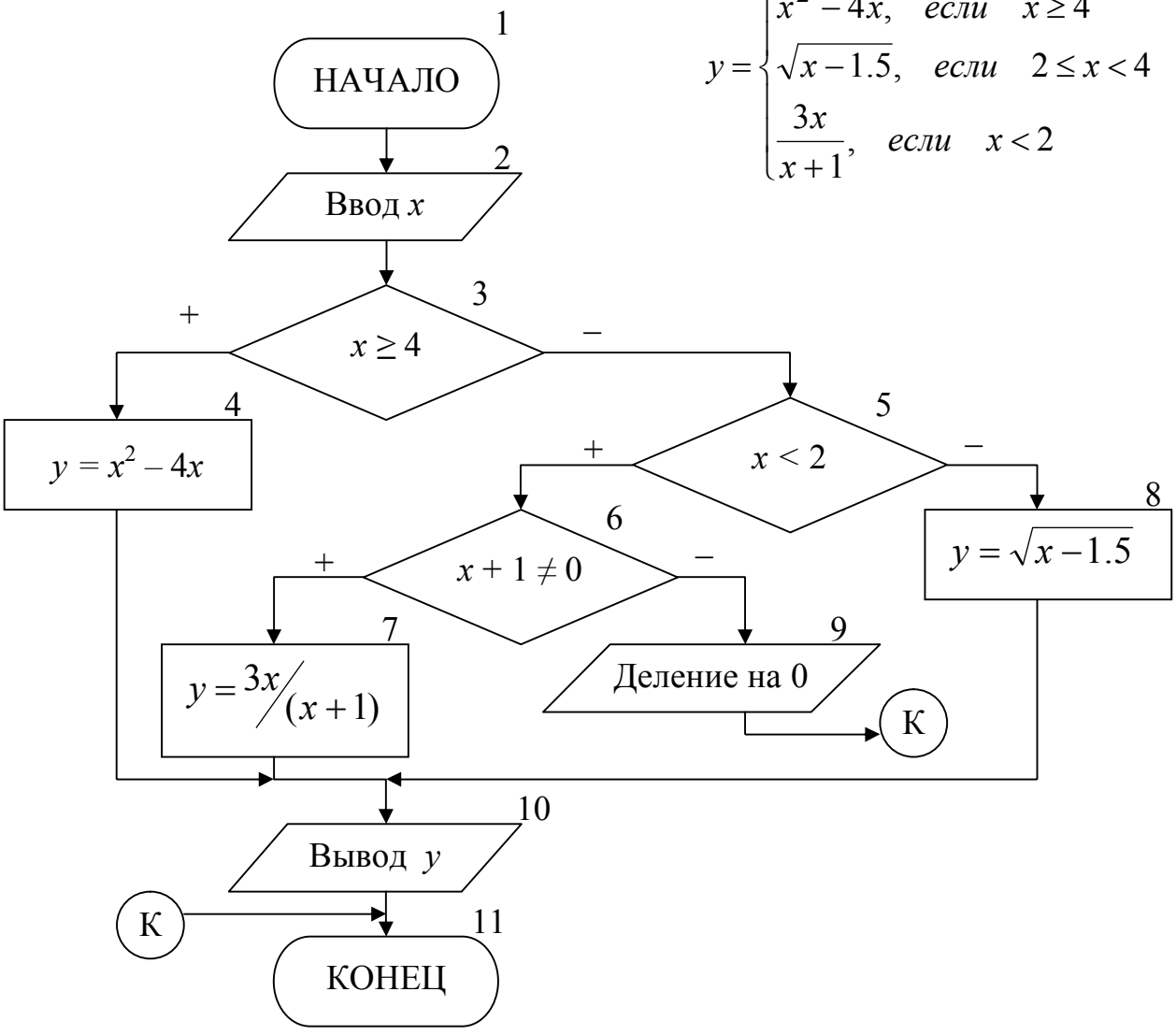


Рисунок 3.3. Алгоритм разветвляющейся структуры

Для выбора формул, по которым будет вычисляться значение ***y***, необходимо проверить три условия. Такие ограничения являются взаимоисключающими усло-

виями и достаточно проверить только два условия, т.е. если не выполняются два условия, то третье выполняется автоматически и его не надо проверять.

Исходными данными для решения поставленной задачи является значение ***x***, которое вводится в блоке 2. Затем в блоке 3 проверяется 1-е ограничение (***x***≥4), в случае его выполнения происходит переход к блоку 4, в котором вычисляется значение ***y*** по 1-й формуле. Если результатом проверки 1-го условия является «ЛОЖЬ» (это означает, что ***x*** < 4), то необходимо проверить одно из двух оставшихся условий. Обычно выбирается более «короткое» ограничение (в блоке 5 проверяется 3-е ограничение). Если ***x*** < 2, то необходимо вычислять ***y*** по 3-й формуле, в которой может возникнуть «аномалия» – деление на 0, поэтому в блоке 6 проверяется неравенство знаменателя дроби нулю. И только после этого в блоке 7 вычисляется значение ***y***. В том случае, когда невозможно вычислить значение ***y*** по 3-й формуле, выводится сообщение о возникновении ошибки (блок 9) и выполняется переход на конец алгоритма. И, наконец, если значение ***x*** не удовлетворяет ни 1-му, ни 3-му ограничению, значит, выполняется 2-е ограничение и ***y*** вычисляется по 2-й формуле (блок 8). Эта формула также содержит «аномалию» – подкоренное выражение должно быть ≥ 0. Однако, проверять её не надо, так как при любом значении ***x,*** попадающем в интервал [2; 4[, подкоренное выражение всегда > 0. Не зависимо от того, по какой формуле будет вычислено значение ***y*** его надо вывести на экран, поэтому выходы блоков 4, 7 и 8 объединяются и происходит переход к блоку 10, в котором выводится ***y***.

За одно выполнение алгоритма может быть вычислено только одно значение ***y*** в зависимости от введенного значения ***x***. Всего же существует 4 варианта работы алгоритма в зависимости от значения ***x***. При этом если возникнет «аномалия», ***y*** вычислен не будет.

4. АЛГОРИТМЫ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

4.1. Структура и основные типы циклов

В **алгоритмах циклической структуры** выполнение одних и тех же действий может повторяться несколько раз. Организация циклического вычислительного процесса выполняется в несколько этапов:

1-й этап – **подготовка к выполнению цикла**. На этом этапе задаются начальные значения для параметра цикла и переменных, использующихся для хра-

нения накапливающихся величин (сумма, количество или произведение вычисляемых величин). **Параметр цикла** – это переменная, на основе которой строится цикл. Она должна удовлетворять трем условиям: являться исходной величиной для выполнения вычислений; изменяться по определенному закону (чаще всего это закон арифметической прогрессии); оказывать влияние на условие завершения повторяющихся вычислений.

2-й этап – **тело цикла**. Оно содержит арифметические и логические действия, которые могут повторяться определенное количество раз. В конце тела цикла обязательно должен быть блок, в котором изменяется значение параметра цикла.

3-й этап – **условие выхода из цикла**, которое предотвращает бесконечное выполнение цикла. С помощью этого условия проверяется надо ли повторять вычисления, или выходить из цикла.

Блок-схема изображающая этапы организации цикла представлена на рис. 4.1.1. Такая организация циклического вычислительного процесса называется **циклом с постусловием**. В этом случае, после каждого выполнения тела цикла, проверяется условие выхода из цикла. Если оно не выполняется, то происходит возврат на начало тела цикла и повторение вычислений. Когда условие выхода будет выполнено, произойдет завершение работы и выход из цикла. Наличие линии возврата в блок-схеме является основным признаком циклического вычислительного процесса.

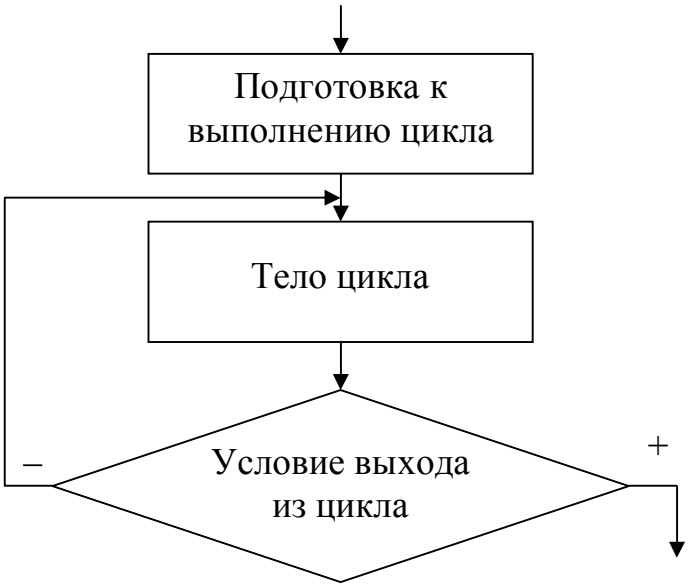


Рисунок 4.1.1. Организация цикла с постусловием

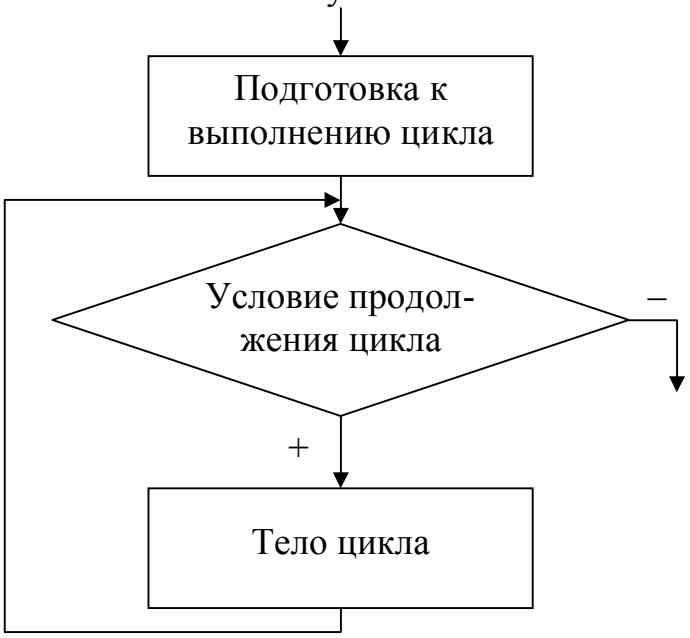


Рисунок 4.1.2. Организация цикла с предусловием

Кроме цикла с постусловием также существуют цикл с предусловием и

цикл «Для».

В **цикле с предусловием** (рис. 4.1.2) перед началом тела цикла проверяется **условие продолжения цикла**. При этом если условие продолжения цикла является истиной, то выполняются действия, составляющие тело цикла, и происходит переход в начало на проверку условия. Цикл завершает свою работу в том случае если условие продолжения цикла не выполняется – становится ложью. Таким образом, в цикле с постусловием в отличие от цикла с предусловием, тело цикла всегда выполнится хотя бы один раз.

Цикл «Для» реализуется на основе блока модификации и представляет собой вариант цикла с предусловием, в котором предполагается, что часть действий по организации цикла выполняется автоматически. Работа цикла «Для» более подробно будет рассмотрена в последующих разделах.

Цикл, в состав которого не входят другие циклы, называется простым.

Пример 4.1. Составить блок-схему алгоритма, вычисляющего значения функции $y = 2 \cdot x + 0.5$, при различных значениях x, принадлежащих интервалу $1 \le x \le 3$, $hx = 0.5$.

Для решения поставленной задачи необходимо организовать перебор всех возможных значений x из заданного интервала с указанным шагом (1; 1.5; 2; 2.5; 3). Для каждого полученного x вычислить значение функции y, т.е. организовать циклический вычислительный процесс. Блок-схема алгоритма, в которой используется цикл с постусловием, приведена на рисунке 4.1.3.

Исходными данными для решения поставленной задачи являются ***xn, xk, hx*** (начальное, конечное значения x из заданного интервала и шаг изменения x), которые вводятся в блоке 2. В нашем случае необходимо будет ввести ***xn*** =1**, *xk*** =3**, *hx*** =0.5.

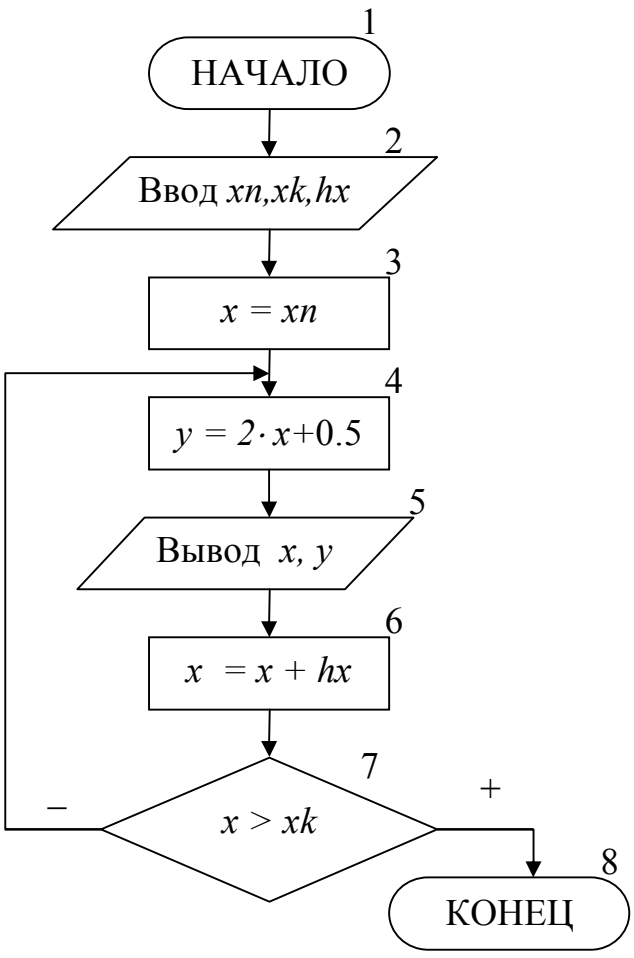


Рисунок 4.1.3. Блок-схема для примера 4.1.

Такая организация ввода исходных данных обеспечивает соответствие блок-схемы двум свойствам алгоритма – определенность и массовость, и в дальнейшем будет называться **составлением алгоритма в «общем виде»** (все исходные данные должны вводиться, а не присваиваться в процессе выполнения алгоритма).

Затем происходит подготовка к выполнению цикла – блок 3, в котором параметру цикла присваивается начальное значение. В качестве параметра цикла выбрана переменная ***x***, удовлетворяющая трем условиям (***x*** является исходным значением для вычислений, изменяется по закону арифметической прогрессии, и оказывает влияние на завершение работы цикла). После этого начинается тело цикла (блоки 4-7). Вычисляется значение ***y*** (блок 4). Рассчитанное значение ***y*** выводится на экран одновременно с соответствующим значением параметра цикла ***x*** (блок 5). Происходит изменение параметра цикла (блок 6), таким образом, получается следующее значение ***x*** из заданного интервала. В конце тела цикла стоит условие выхода из цикла ***x > xk*** (блок 7), с помощью которого проверяется, не вышло ли новое значение ***x*** за правую границу заданного интервала. Если значение ***x*** не превышает ***xk***, то происходит возврат на начало тела цикла (блок 4) и повторяются те же самые действия. В случае выполнения условия происходит выход из цикла (переход в блок 8) и алгоритм завершает свою работу.

Однократное выполнение тела цикла называется **шагом**. За один шаг вычисляется одно значение результирующей функции при одном (текущем) значении параметра цикла. Результаты пошагового выполнения цикла, приведенного на рис. 4.1.3, представлены в таблице 4.1.

Таблица 4.1. Пошаговое выполнение цикла

№ шага	Текущее значение ***x*** (в начале цикла)	Вычисленное значение ***y***	Новое значение ***x*** (в конце цикла)	Результат проверки условия выхода из цикла (***x > xk***)
1	***1*** (***xn)***	2,5	1,5	Ложь
2	1,5	3,5	2	Ложь
3	2	4,5	2,5	Ложь
4	2,5	5,5	3	Ложь
5	***3*** (***xk***)	6,5	***3,5***	Истина

Количество шагов выполнения цикла ***N*** можно вычислить до начала работы алгоритма по следующей формуле (1):

$$N = \left]\frac{xk - xn}{hx}\right[+ 1 \qquad (1)$$

где] [обозначают целую часть выражения.

Циклические вычислительные процессы, для которых можно вычислить количество шагов цикла без выполнения алгоритма, называются **циклами с известным числом повторений**. Для реализации циклов с известным числом повторений можно равноценно использовать любой из трех стандартных типов цикла (с постусловием, с предусловием, «Для»).

Если в цикле при выполнении вычислений может возникнуть «**аномалия**», то завершать выполнение алгоритма, как в разветвляющемся вычислительном процессе, не надо. В этом случае необходимо пропустить все операции, использующие переменную, значение которой невозможно вычислить, и выполнить переход к блоку, в котором изменяется значение параметра цикла, т.е. продолжить работу цикла. При следующем значении параметра цикла «аномалия» может не возникнуть, и работа цикла пойдет естественным путем.

4.2. Алгоритмы нахождения суммы, произведения и количества вычисленных значений

Часто наряду с циклическим вычислением значений величины или совокупности величин необходимо определить сумму, произведение или количество всех получаемых значений или некоторых из них.

При выполнении суммирования и вычислении произведения или количества рационально использовать принцип постепенного накапливания значений величин, получаемых на каждом шаге цикла. Для хранения таких «накапливающихся» величин используются дополнительные переменные, которым предварительно необходимо задать начальные значения. Обычно это делается на этапе подготовки к выполнению цикла (см. п. 4.1). **В качестве начального значения для суммы и количества используется ноль, для произведения – единица.**

Для пояснения принципа вычисления накапливающихся величин рассмотрим пример аналогичный примеру из п. 4.1.

Пример 4.2. Составить блок-схему алгоритма, вычисляющего значения функции $y = 2 \cdot x + 1$, при различных значениях параметра x, принадлежащих интервалу $-3 \le x \le 3$, $hx = 1$. В качестве дополнительной задачи необходимо найти:

количество (***k***) вычисленных значений ***y***, которые ≤ 0; сумму (***S***) всех значений ***y*** (***S*** **= Σ*y***) и произведение (***P***) значений ***y***, которые > 0 ($P = \prod_{y>0} y$).

При решении этой задачи необходимо организовать перебор всех возможных значений ***x*** из заданного интервала с указанным шагом, и для каждого полученного ***x*** вычислить значение функции ***y***. Разрабатываемый алгоритм является циклом с известным числом повторений, так как количество повторений вычислений можно определить по формуле (1):

$$N = \left\lceil \frac{3-(-3)}{1} \right\rceil + 1 = 7$$

Поэтому для реализации алгоритма, в отличие от примера 4.1, можно воспользоваться циклом с предусловием, блок-схема которого приведена на рис. 4.2.

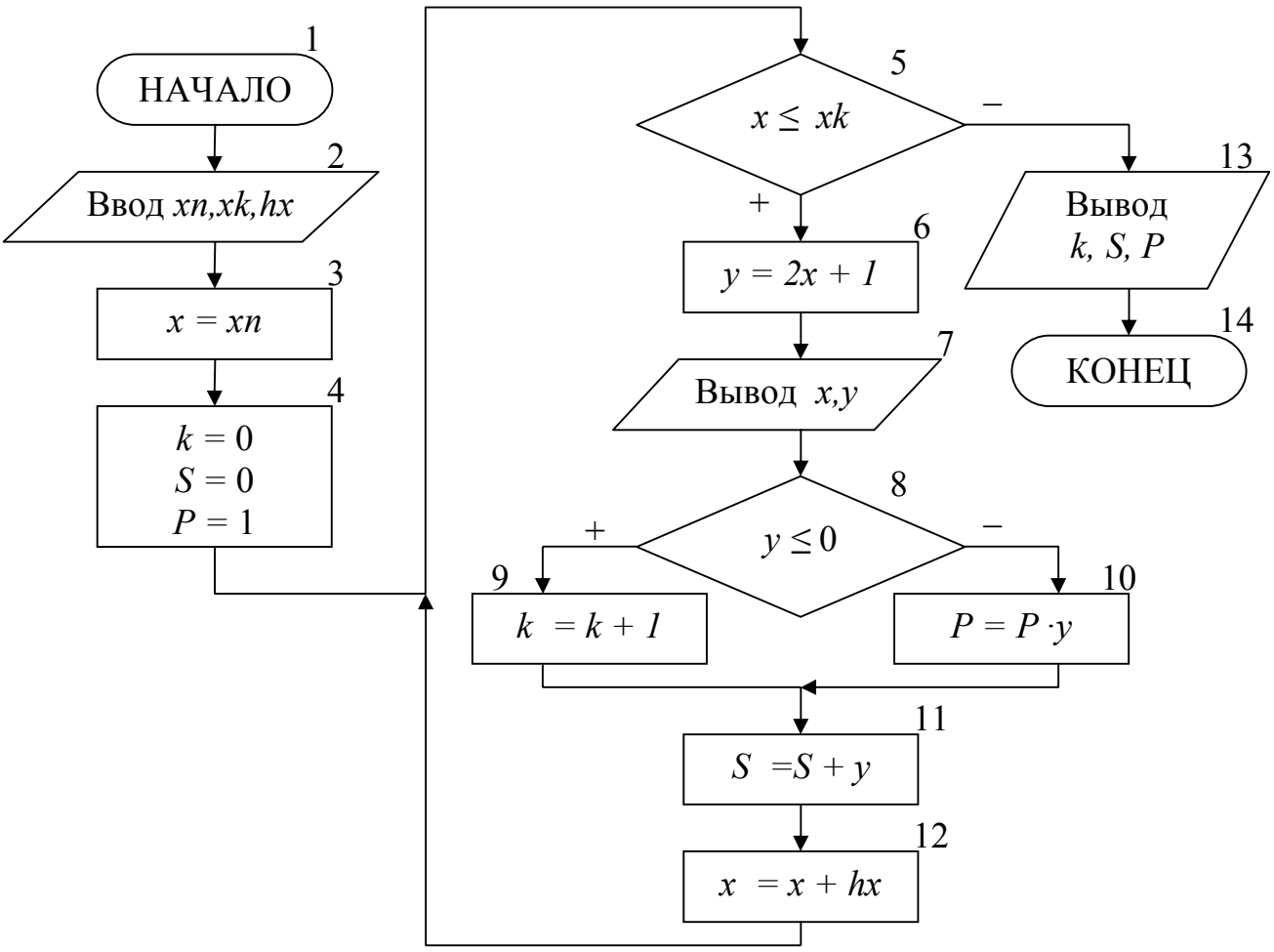


Рисунок 4.2. Блок-схема алгоритма для примера 4.2.

Исходными данными, как и в примере 4.1, являются ***xn, xk, hx***, которые вводятся в блоке 2. В нашем случае необходимо будет ввести ***xn*** =-3**,** ***xk*** =3**,** ***hx*** =1. На этапе подготовки к выполнению цикла задаются начальные значения для па-

раметра цикла ***x*** (блок 3) и дополнительных величин ***k, S, P*** (блок 4). Затем начинается тело цикла с предусловием (блоки 5-12).

При входе в цикл проверяется условие продолжения вычислений в теле цикла (блок 5). Вычисляется (блок 6) и выводится на экран (блок 7) значение ***y*** при текущем значении параметра цикла ***x***. Затем проверяется вычисленное значение ***y*** (блок 8), и если $y \le 0$, то ***k*** увеличивается на единицу (блок 9), в противном случае текущее значение ***y*** учитывается в произведении ***P*** (блок 10). Кроме этого, каждое вычисленное значение ***y*** добавляется к сумме ***S*** (блок 11). В конце тела цикла выполняется изменение параметра цикла (блок 12), т. е. вычисляется следующее значение ***x*** из заданного интервала, и происходит возврат в начало цикла. После выхода из цикла выводятся значения переменных ***k, P, S*** (блок 13).

Результаты пошагового выполнения цикла в алгоритме, приведенном на рис. 4.2, представлены в таблице 4.2.

Таблица 4.2. Пошаговое выполнение цикла

№ шага	Текущее значение ***x*** (в начале цикла)	Результат проверки условия продолжения цикла (***x*** ≤ ***xk***)	Вычисл. значение ***y***	Текущие значения накапливающихся величин ***k, P, S***	Новое значение ***x*** (в конце цикла)
1	***-3*** (***xn)***	Истина	-5	***k***=1, ***P***=1, ***S***=-5	-2
2	-2	Истина	-3	***k***=2, ***P***=1, ***S***=-8	-1
3	-1	Истина	-1	***k***=3, ***P***=1, ***S***=-9	0
4	0	Истина	1	***k***=3, ***P***=1, ***S***=-8	1
5	1	Истина	3	***k***=3, ***P***=3, ***S***=-5	2
6	2	Истина	5	***k***=3, ***P***=15, ***S***=0	3
7	***3*** (***xk***)	Истина	7	***k***=3, ***P***=105, ***S***=7	4
8	***4***	Ложь			

Пошаговое выполнение цикла показывает, что итоговые значения суммы, количества и произведения (***k=3, P=105, S=7***) будут получены после завершения работы цикла. Поэтому они должны выводиться **только после выхода из цикла**. Если вывод ***k, P, S*** организовать внутри тела цикла, то будут выведены все промежуточные значения этих величин, приведенные в таблице. Также будет неправильно поставить вывод значения параметра цикла ***x*** и вычисляемой величины ***y*** (блок 7) после выхода из цикла. Из таблицы 4.2 видно, что в этом случае будут выведены последние значения, хранящиеся в этих переменных (***x***=4 и ***y***=7).

4.3. Циклы с неизвестным числом повторений

При решении некоторых задач может возникнуть ситуация, когда невозможно заранее определить количество повторений вычислений. В таких случаях речь идет о **циклах с неизвестным числом повторений**.

В циклах с неизвестным числом повторений вычислительный процесс завершается при выполнении некоторого дополнительного условия. Значения параметра цикла уже не задаётся в виде диапазона, а только указывается его начальное значение и шаг изменения. Тем не менее, организация цикла выполняется по стандартной методике, указанной в п. 4.1. Отличие заключается в том, что не любой тип циклического вычислительного процесса можно использовать. Тип цикла определяется в соответствии с заданным дополнительным условием завершения вычислений. Это однозначно исключает возможность использование цикла «Для» на основе блока модификации. Для определения количества шагов повторения цикла необходимо организовать счетчик.

Пример 4.3. Составить блок-схему алгоритма, вычисляющего значения функции $y = ln(1 + 0.2 \cdot x)$, при различных значениях параметра x, изменяющегося от $xn \le 3$ с шагом $hx = -2$. При этом вычислять y до тех пор, пока выражение под знаком логарифма остается больше 0. Определить количество (k) вычисленных значений y.

Перед решением задачи необходимо определить тип цикла, который будет использоваться. Вычисляемое выражение содержит «аномалию» (значение под знаком логарифма должно быть больше 0), которая в тоже время является условием завершения вычислений. Другими словами, необходимо вначале проверить возможность вычисления значения y, а только затем вычислять его. Поэтому в данном случае для организации вычислительного процесса можно использовать только цикл с предусловием (рис. 4.3).

В качестве исходных данных вводятся значения переменных xn, hx (блок 2). В блоке 3 задаются начальные значения для параметра цикла x и счетчика количества повторений цикла k. После этого, в блоке 4 одновременно проверяются условие продолжения цикла и возможная «аномалия» (выражение под знаком логарифма должно быть больше 0). В теле цикла для текущего значения x вычисляется и выводится соответствующее значение y (блоки 5-6), а также увеличивается значение счетчика вычисленных y (блок 7). В конце тела цикла выполняется пере-

ход к следующему значению параметра цикла ***x*** (блок 8). Цикл работает до тех пор, пока под знаком логарифма не появится выражение ≤ 0. После выхода из цикла в блоке 9 выводится переменная ***k*** – количество вычисленных значений ***y***.

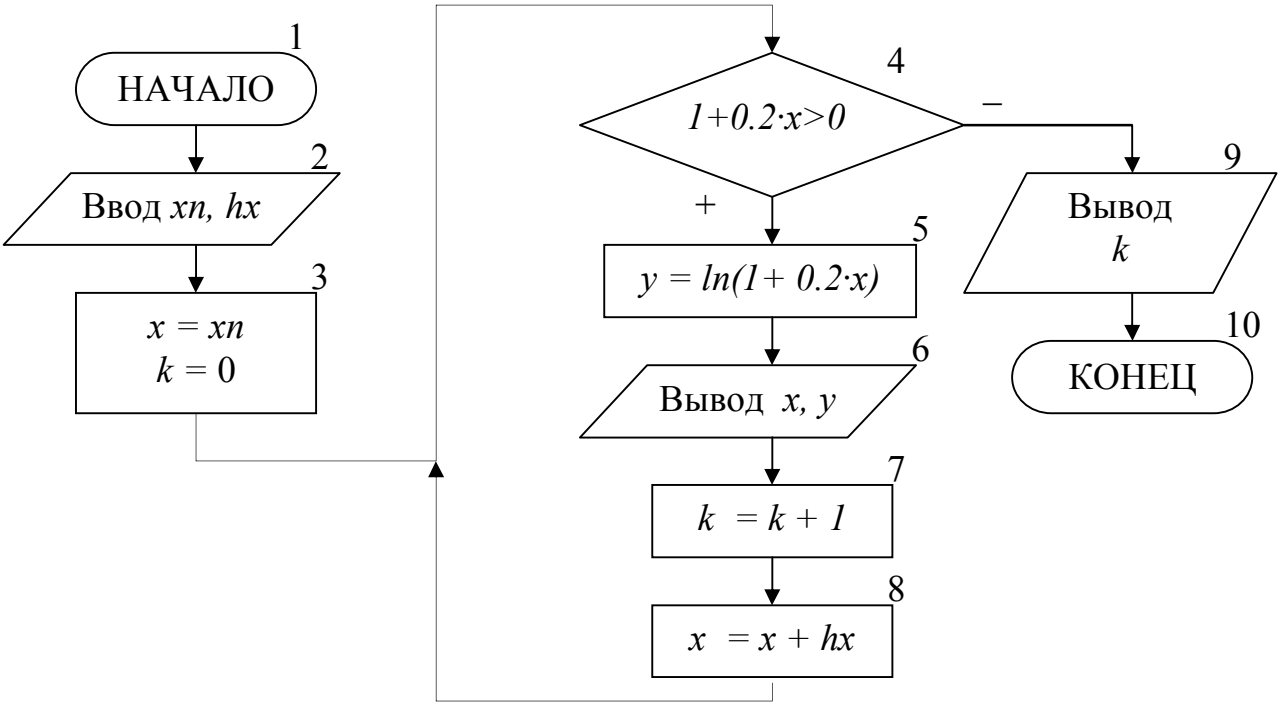


Рисунок 4.3. Блок-схема алгоритма для примера 4.3.

Результаты пошагового выполнения цикла в алгоритме, приведенном на рис. 4.3, представлены в таблице 4.3.

Таблица 4.3. Пошаговое выполнение цикла

№ шага	Текущее значение ***x*** (в начале цикла)	Результат проверки условия продолжения цикла (***1+0.2·x>0***)	Вычисл. значение ***y***	Текущие значения ***k***	Новое значение ***x*** (в конце цикла)
1	***3*** (***xn***)	Истина (*1+0.2·3=1.6 >0*)	0.47	***k***=1	1
2	1	Истина (*1+0.2·1=1.2 >0*)	0.18	***k***=2	-1
3	-1	Истина (*1+0.2·-1=0.8 >0*)	-0.22	***k***=3	-3
4	-3	Истина (*1+0.2·-3=0.4 >0*)	-0.92	***k***=4	-5
5	***-5***	Ложь (*1+0.2·-5=0 >0*)			

4.4. Вложенные циклы

Решение некоторых задач требует выполнения перебора значений не одной, а нескольких величин одновременно. В этом случае речь идет о применении **вложенных циклов**, каждый из которых организовывается по стандартному принци-

пу (может быть любого из трех типов) и осуществляет перебор только одного параметра. При этом первый цикл называется внешним, а вложенные в него – внутренними. Причем один и тот же цикл может быть внешним по отношению к одному и внутренним по отношению к другому циклу. Границы внутреннего цикла не могут выходить за границы внешнего по отношению к нему цикла.

Для каждого значения параметра внешнего цикла происходит перебор всех возможных значений параметра внутреннего цикла. Другими словами, всегда выполняется в первую очередь самый внутренний цикл. Такая организация циклов дает возможность перебрать значения их параметров во всех возможных комбинациях.

Пример 4.4. Составить блок-схему алгоритма, вычисляющего значения функции $y = a - 2 \cdot b$, для всех возможных комбинаций значений параметров a и b, принадлежащих интервалам $0 \le a \le 2$, $ha = 1$ и $-2 \le b \le 2$, $hb = 2$, соответственно. Определить среднее арифметическое (S) вычисленных значений y.

Для решения поставленной задачи необходимо организовать два вложенных цикла по перебору параметров a и b. При этом каждый цикл является циклом с известным числом повторений, т. к. заранее можно вычислить количество значений a из заданного интервала ($Na =](2 - 0)/1[+ 1 = 3$) и количество значений b из заданного интервала ($Nb =](2 - (-2))/2[+ 1 = 3$), а следовательно и общее количество вычисленных значений y ($N = Na \cdot Nb$). Значит, для организации циклов можно использовать любой из трех стандартных типов.

Кроме этого y одновременно зависит от двух параметров a и b, поэтому не имеет принципиального значения по какому параметру делать внешний цикл, а по какому – внутренний (внешним может быть цикл по параметру a, а внутренним – по параметру b, и наоборот). В случае же если одна из вычисляемых величин зависит только от одного параметра, то цикл по этому параметру рационально сделать внешним и вычислять величины, зависящие только от этого параметра во внешнем цикле. Такая организация позволит избежать многократного вычисления одних и тех же значений во внутреннем цикле.

На рисунке 4.4 представлена блок-схема алгоритма, в которой для решения поставленной задачи, используется два вложенных цикла: внешний цикл с предусловием по параметру a и внутренний цикл с постусловием по параметру b.

Работа алгоритма начинается с ввода исходных данных (блок 2), которые представляют собой начальные, конечные значения и шаг изменения параметров

a и ***b*** из заданных интервалов. На этапе подготовки к выполнению внешнего цикла (блок 3) параметру цикла ***a*** присваивается начальное значение ***an***, а также обнуляются переменные ***S*** и ***k***, используемые для хранения суммы и количества вычисленных значений ***y***.

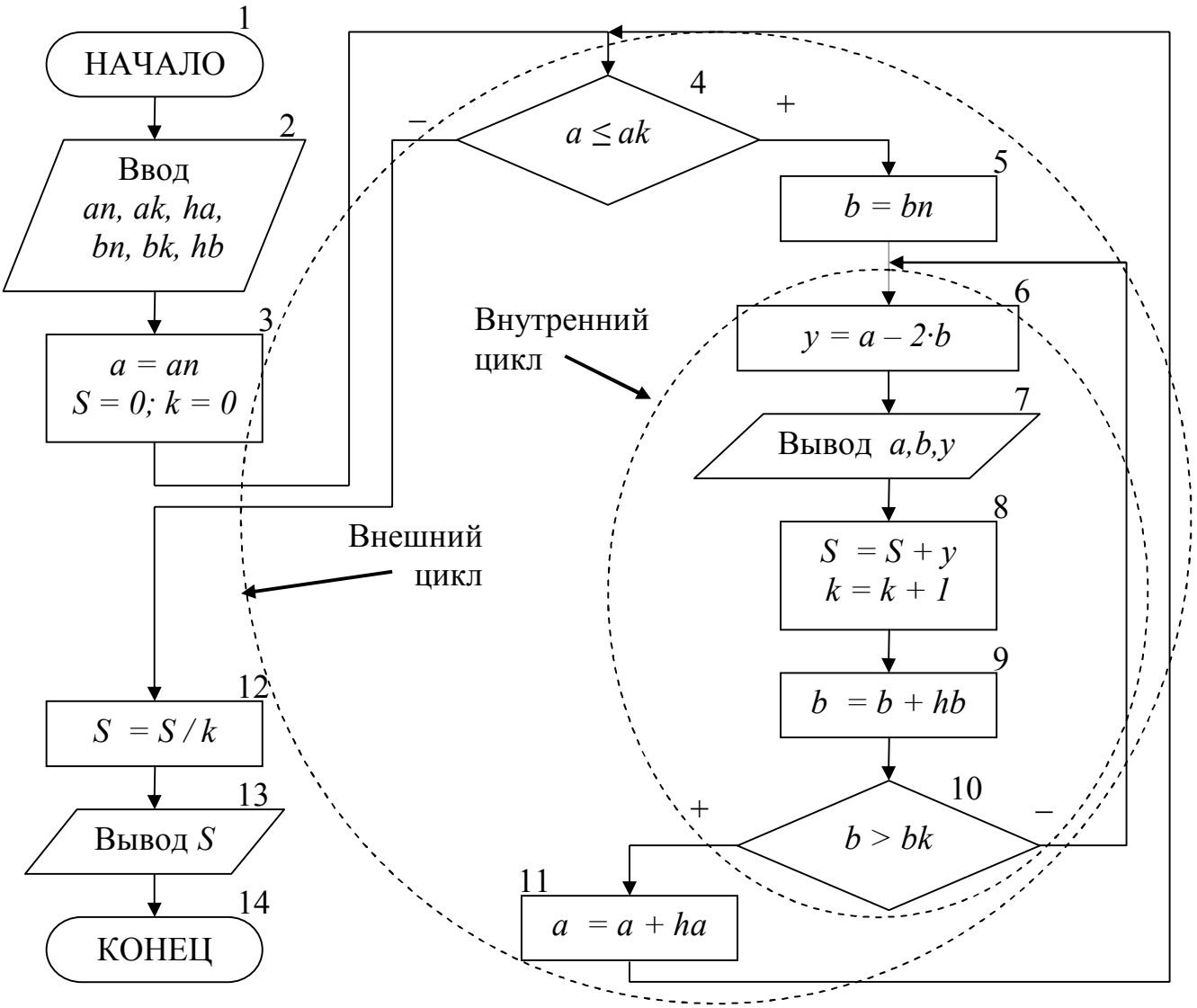


Рисунок 4.4. Блок-схема алгоритма для примера 4.4.

Тело внешнего цикла с предусловием включает блоки 4-11. Если текущее значение ***a*** не выходит за правую границу интервала (блок 4), то происходит выполнение тела внешнего цикла, которое включает в себя внутренний цикл. В блоке 5 происходит подготовка к выполнению внутреннего цикла, его параметру ***b*** присваивается начальное значение ***bn***.

Тело внутреннего цикла с постусловием включает блоки 6-10. В блоке 6 вычисляется текущее значение ***y***, а в блоке 7 происходит вывод вычисленного ***y*** и соответствующих ему значений ***a*** и ***b***. В блоке 8 полученное значение ***y*** добавляется к сумме ***S*** и увеличивается значение счетчика вычисленных ***y***. Затем проис-

ходит изменение параметра внутреннего цикла ***b*** на значение шага ***hb*** (блок 9) и выполняется проверка условия выхода из внутреннего цикла (блок 10).

Если текущее значение ***b*** не превышает конечного ***bk***, то происходит возврат на начало внутреннего цикла и повторяется вычисление ***y*** при новом значении ***b***. В тоже время значение параметра внешнего цикла ***a*** остается неизменным. Таким образом, при одном значении параметра внешнего цикла происходит перебор всех значений параметра внутреннего цикла.

После выхода из внутреннего цикла в блоке 11 происходит изменение параметра внешнего цикла ***a*** на значение шага ***ha***. Затем выполняется переход на начало цикла (блок 4) и повторяются описанные выше действия, пока не завершит работу внешний цикл. После выхода из внешнего цикла в блоках 12-13 вычисляется и выводится ***S -*** среднее арифметическое вычисленных значений ***y***.

Результаты пошагового выполнения циклов в алгоритме, приведенном на рис. 4.4, представлены в таблице 4.4.

Таблица 4.4. Пошаговое выполнение алгоритма с вложенными циклами

№ шага	Текущее знач. ***a*** (в начале цикла)	Проверка условия продолжения внешнего цикла (***a*** ≤ ***ak***)	Текущее знач. ***b*** (в начале цикла)	Вычисл. значение ***y***	Текущие значения ***k, S***	Новое знач. ***b*** (в конце цикла)	Проверка условия выхода из внутреннего цикла (***b*** > ***bk***)	Новое знач. ***a*** (в конце цикла)
1	0 (***an)***	Истина	-2 (***bn)***	4	***k***=1, ***S***=4	0	Ложь	-
2	-	-	0	0	***k***=2, ***S***=4	2	Ложь	-
3	-	-	2 (***bk)***	-4	***k***=3, ***S***=0	4	Истина	1
4	1	Истина	-2 (***bn)***	5	***k***=4, ***S***=5	0	Ложь	-
5	-	-	0	1	***k***=5, ***S***=6	2	Ложь	-
6	-	-	2(***bk)***	-3	***k***=6, ***S***=3	4	Истина	2
7	2 (***ak)***	Истина	-2 (***bn)***	6	***k***=7, ***S***=9	0	Ложь	-
8	-	-	0	2	***k***=8,***S***=11	2	Ложь	-
9	-	-	2(***bk)***	-2	***k***=9, ***S***=9	4	Истина	3
10	3	Ложь	-	-	-	-	-	-

Следует также обратить внимание на то, что начальное значение параметра внутреннего цикла ***b*** необходимо задавать каждый раз перед началом выполнения вложенного цикла (блок 5), т.е. внутри внешнего цикла. Если это будет сделано

одновременно с заданием начального значения для параметра внешнего цикла ***a*** (блок 3), то внутренний цикл выполнится только при начальном значении ***a***. На этом алгоритм завершит свою работу и, следовательно, не будут перебраны все возможные комбинации значений ***a*** и ***b***.

5. АЛГОРИТМЫ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ

Массив – это последовательность однотипных элементов, каждый из которых имеет одно и тоже имя, но однозначно определяется своим номером (**индексом**). Массив – это не скалярная величина, а структурированный тип данных.

Обычно используются одномерные (вектор) и двумерные (матрица) массивы. В одномерном массиве каждый элемент имеет один индекс, определяющий положение элемента в массиве. В двумерном массиве каждый элемент имеет два индекса. Чаще всего нумерация индексов начинается с 1, но может и с 0.

	X_1	X_2	X_3	…	X_N
Значение элемента	2	0	1	…	7
Индекс элемента i	1	2	3	…	N

Основными характеристиками массива являются:

– размерность, т.е. количество элементов (обычно обозначается N);

– значения элементов (например, $X_1 = 2$; $X_3 = 1$ и т.д.).

Обработка массива обычно заключается в последовательном переборе его элементов и выполнении над ними однотипных операций, т.е. обработка массива является циклическим вычислительным процессом. Для этого достаточно организовать цикл по перебору индексов элементов массива. Наиболее рационально для этой цели использовать цикл «Для» на основе блока модификации (рис. 5.1).

При входе в блок модификации (рис. 5.1.а,б, линия 1) автоматически выполняются следующие действия. Параметру цикла ***i*** присваивается начальное значение ***in*** и проверяется, не превышает ли оно конечного значения ***ik***. Если результатом проверки условия является истина, то происходит переход к выполнению тела цикла (линия 2). После этого осуществляется возврат в блок модификации (линия 3), увеличение параметра цикла ***i*** на значение шага ***hi*** и проверка условия продолжения цикла и т.д. Когда текущее значение параметра цикла ***i*** превысит конечное значение ***ik***, цикл завершит свою работу (линия 4). Если шаг изменения параметра цикла ***hi*** равен 1, то его в блоке модификации можно не указывать (рис. 5.1,в).

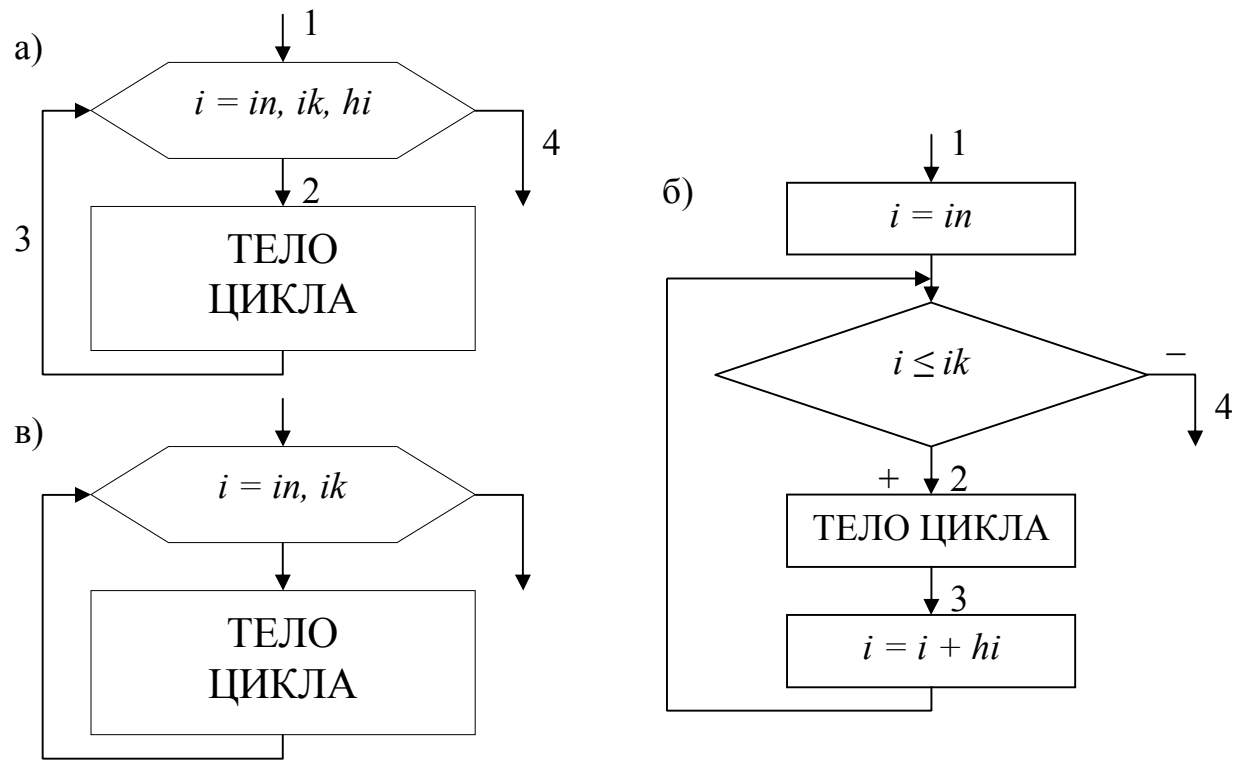


Рисунок 5.1. Цикл «Для» на основе блока модификации (а, в) и пояснение принципа его работы на основе цикла с предусловием (б).

В одних языках программирования, таких как Турбо Паскаль, Delphi, существуют ограничения на использование цикла «Для» (параметр цикла, его начальное и конечное значения должны быть только целочисленного типа; шаг изменения может быть равен только 1 или -1). В других языках программирования, таких как Си, Visual Basic, этих ограничений нет. Но в любом случае цикл «Для» идеально подходит для организации обработки массивов.

5.1. Ввод и вывод элементов одномерного массива

Ввод элементов массива выполняется в два этапа. Вначале указывается его размерность (количество элементов), а затем задаются значения для каждого элемента массива. Рассмотрим два способа организации ввода элементов одномерного массива.

1 способ. Алгоритм ввода массива X размерностью N (i=1÷N) (рис. 5.1.1.а). На первом этапе вводится размерность массива N (блок 1). На втором этапе организовывается цикл «Для» на основе блока модификации, выполняющий поэлементный ввод произвольных однотипных значений компонент массива.

Параметр цикла i в тоже время является индексом элементов массива X.

Блок модификации (блок 2) перебирает значения ***i*** от 1 до ***N*** и для каждого значения ***i*** выполняется ввод значения для X_i-го элемента массива (блок 3), т.е. при ***i*** =1 будет введено значение для X_1, при ***i*** =2 – для X_2 и т.д. Таким образом, за один шаг выполнения цикла вводится один элемент массива X.

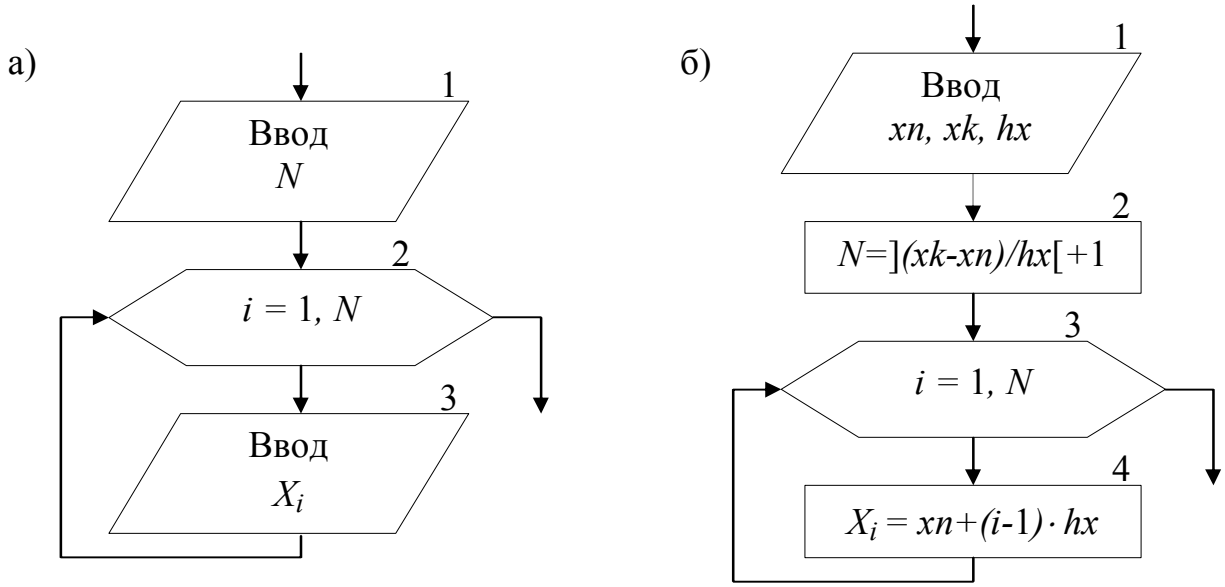


Рисунок 5.1.1. Поэлементный ввод массива (а) и вычисление элементов массива, значения которых лежат в заданном интервале (б).

2 способ. Алгоритм формирования массива X, значения элементов которого принадлежат заданному интервалу $xn \le X_i \le xk$ с шагом изменения ***hx*** (рис. 5.1.1.б). На первом этапе вводятся значения ***xn***, ***xk*** и ***hx*** (блок 1) и вычисляется ***N*** – размерность массива X (блок 2). На втором этапе организовывается цикл «Для» на основе блока модификации (блок 3), в котором перебираются значения ***i***. На каждом шаге цикла вычисляется значение одного элемента массива X (блок 4), т.е. при ***i*** =1 будет вычислено значение для $X_1=xn+(1-1)\cdot hx=xn$, при ***i*** =2 – для $X_2=xn+(2-1)\cdot hx=xn+hx$ и т.д.

Вывод массива также выполняется поэлементно с помощью цикла «Для» (рис. 5.1.2).

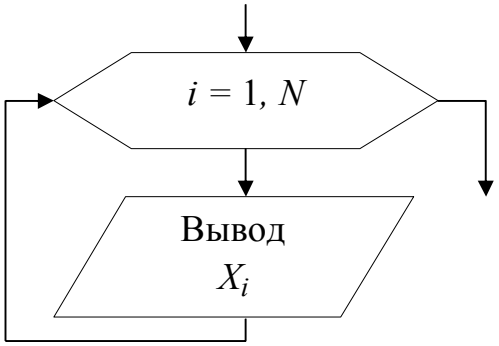


Рисунок 5.1.2. Вывод массива

Циклы по вводу или выводу элементов массивов не обязательно делать отдельно. Их можно объединять с циклами по обработке элементов массивов.

Пример 5.1. Используя элементы исходного массива X размерности ***N***, вычислить элементы массива Y по следующей формуле: $Y_i=1/X_i$. Определить сред-

нее арифметическое положительных элементов массива Y.

Результирующий массив Y будет иметь туже размерность, что и массив X.

Решение поставленной задачи можно разделить на 4 этапа: ввод элементов исходного массива X, вычисление элементов массива Y, вычисление среднего арифметического положительных элементов массива Y и вывод элементов массива Y. Для реализации каждого этапа необходимо организовать отдельные циклы для перебора элементов массивов. Однако, такой способ решения не рационален, хотя и возможен. Вследствие того, что организация всех четырех циклов будет полностью идентична, этапы решения со 2-го по 4-й можно реализовать с помощью только одного цикла «Для». Это значительно уменьшит размер алгоритма и скорость его выполнения. Блок-схема алгоритма приведена на рис. 5.1.3.

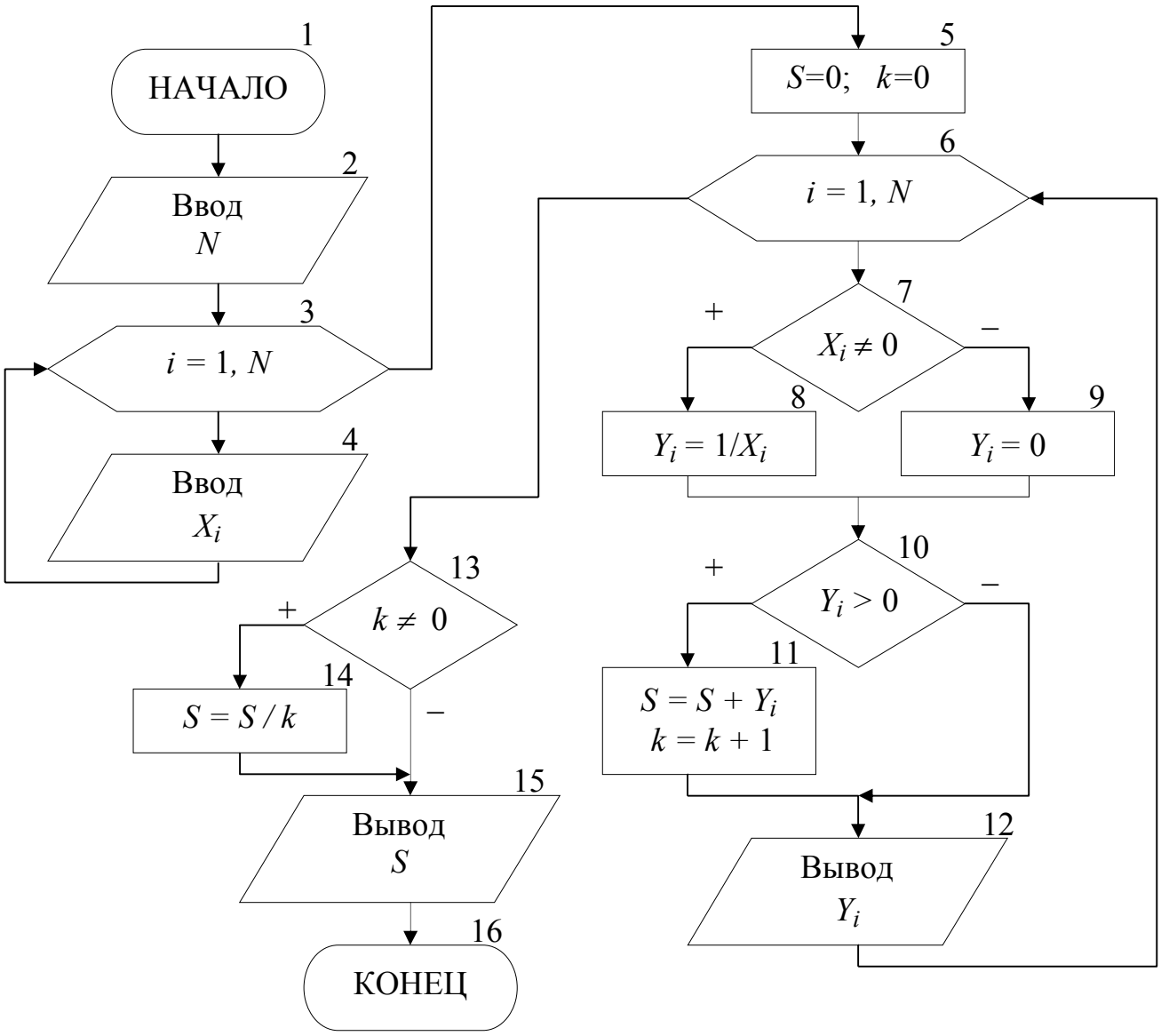


Рисунок 5.1.3. Блок-схема алгоритма для примера 5.1.

Ввод исходного массива (блоки 2-4) осуществляется поэлементно, описанным выше способом. В блоке 5 присваиваются начальные значения для перемен-

ных ***S*** и ***k*** (сумма и количество положительных элементов массива Y). Затем организовывается цикл «Для» (блок 6), на каждом шаге которого будут выполняться следующие действия: вычисление по заданной формуле текущего элемента массива Y (блоки 7-9); проверка – не является ли вычисленный элемент Y_i положительным (блок 10); добавление положительного значения Y_i к сумме ***S*** и увеличение счетчика таких элементов ***k*** на 1 (блок 11); вывод текущего элемента массива Y (блок 12).

Особое внимание следует обратить на наличие возможной «аномалии» в вычисляемом выражении (X_i не должно быть равным 0). **Обработка «аномалий»**, возникающих при вычислении элементов массива, немного отличается от предложенной в п. 4.1. Если просто пропустить все операции, которые невозможно выполнить, то текущий элемент массива не будет вычислен. В результате может быть сформирован массив, у которого часть элементов будет не определено (массив с «дырами»). Дальнейшее использование такого массива недопустимо. Поэтому в случае возникновения аномалии при вычислении текущего элемента массива, ему надо присвоить такое значение, которое не повлияет на дальнейшие вычисления, например 0 (блок 9).

Таким образом, с помощью одного цикла «Для» будут поочередно перебраны все элементы массива X, для каждого элемента X_i вычислен и выведен соответствующий элемент массива Y_i. После выхода из цикла вычисляется и выводится среднее арифметическое значение положительных элементов массива Y (блоки 13-15). При этом предварительно в блоке 13 проверяется возможность деления на ноль.

5.2. Нахождение максимального и минимального элементов массива

Принцип поиска максимального или минимального элемента массива заключается в следующем: в дополнительную переменную заносится значение первого элемента массива, которое принимается за максимум (минимум); затем организовывается перебор оставшихся элементов массива, каждый из которых сравнивается с максимумом (минимумом); если текущий элемент массива оказывается больше (меньше), чем принятый за максимум (минимум), то этот элемент становится максимальным (минимальным). Таким образом, после завершения перебора

элементов массива в дополнительной переменной окажется максимальное (минимальное) значение среди элементов массива. Фрагмент блок-схемы алгоритма, реализующий описанный метод приведен на рисунке 5.2.1.

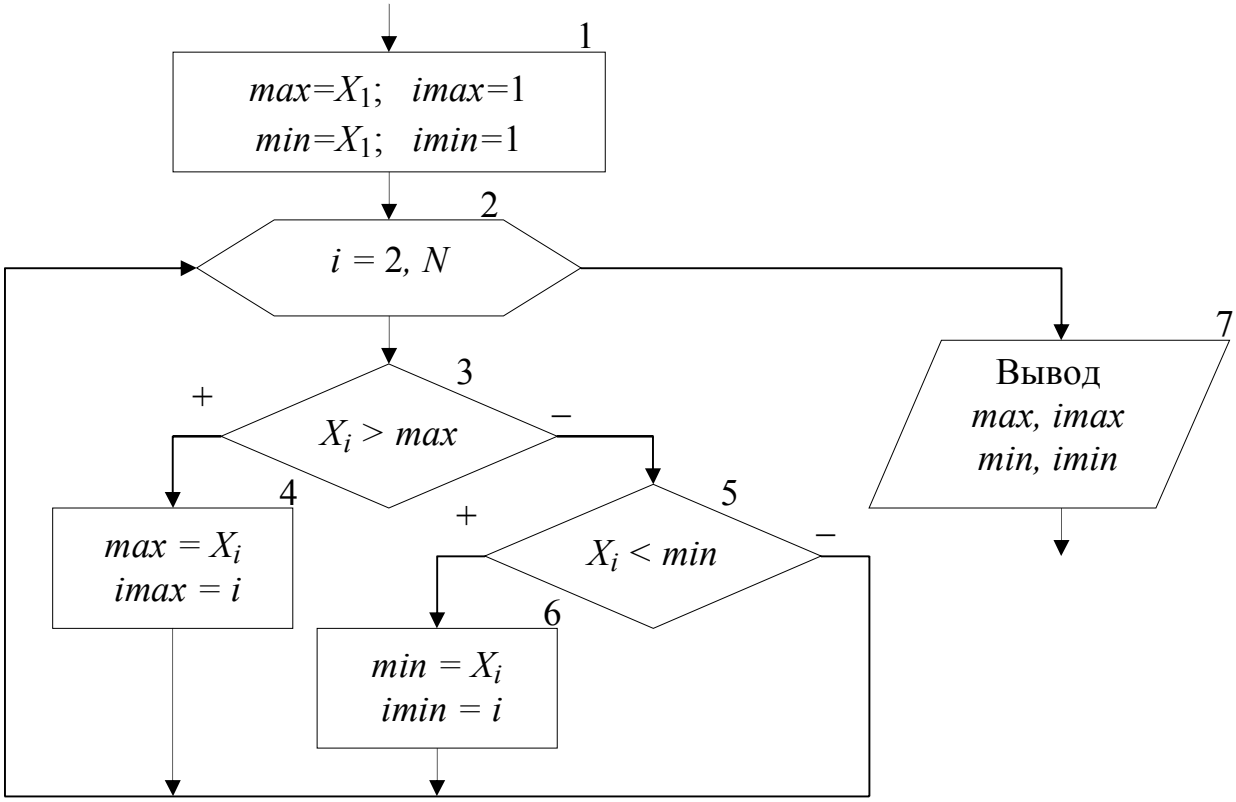


Рисунок 5.2.1. Поиск максимума (минимума) в массиве

В блоке 1 дополнительным переменным ***max*** и ***min***, в которых будут храниться максимальное и минимальное значения соответственно, присваивается значение 1-го элемента массива. Кроме этого введены еще две переменные ***imax*** и ***imin***, которые будут использоваться для хранения номеров максимального и минимального элементов массива. С помощью блока модификации (блок 2) организовывается цикл по перебору элементов массива X (перебор начинается со 2-го элемента, так как 1-й элемент уже обработан в блоке 1). Если текущий элемент массива $\boldsymbol{X_i}$ больше значения, которое хранится в переменной ***max*** (блок 3), то за максимум принимается значение этого элемента: $\boldsymbol{max=X_i}$, и запоминается его номер: ***imax=i*** (блок 4). Если текущий элемент массива не больше максимума, то проверяется, не меньше ли он минимума (блок 5). В случае если текущий элемент $\boldsymbol{X_i}$ окажется меньше минимального значения, которое хранится в переменной ***min***, то за минимум принимается значение этого элемента: $\boldsymbol{min=X_i}$, и запоминается его номер: ***imin=i*** (блок 6). После выхода из цикла будут выведены значения максимального и минимального элементов, а также их номера в массиве (блок 7).

Приведенный выше алгоритм имеет один недостаток – в нем используется две лишние переменные ***max*** и ***min***. Зная индекс элемента массива всегда можно получить его значение, поэтому нет необходимости хранить максимальное и минимальное значения в отдельных переменных. Оптимальный алгоритм приведен на рис. 5.2.2.

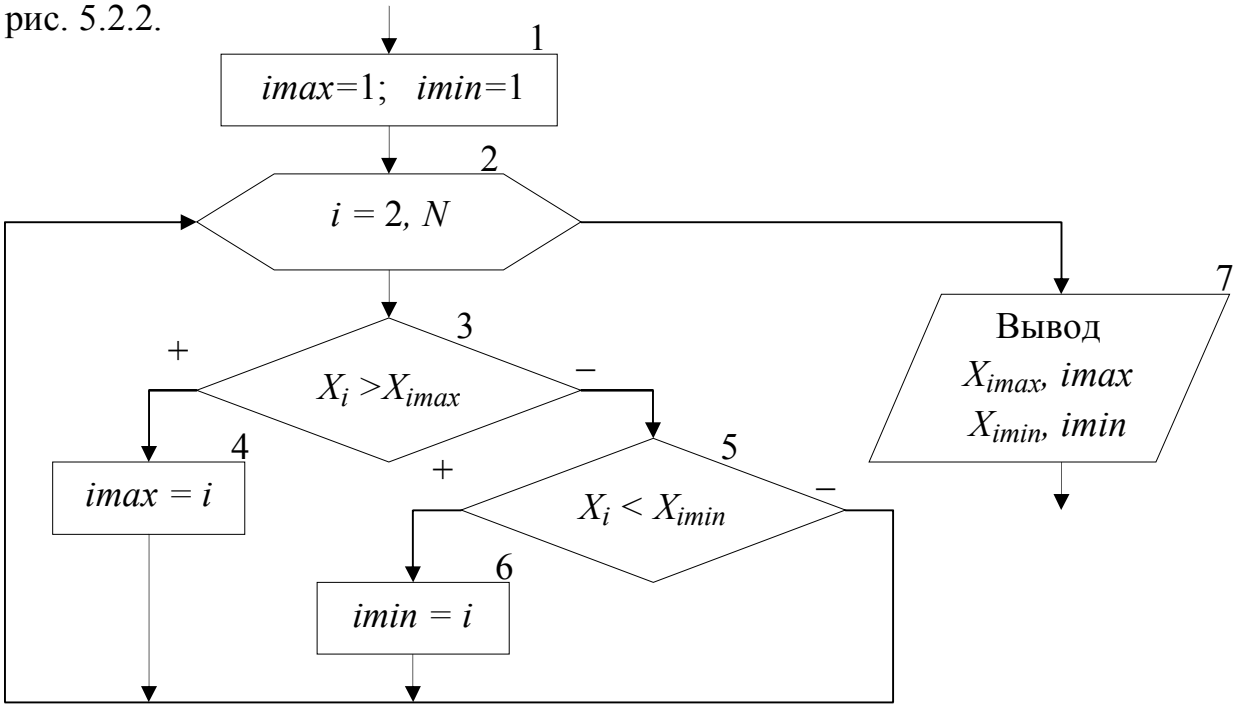


Рисунок 5.2.2. Оптимизация поиска максимума (минимума) в массиве

В блоке 1 дополнительным переменным ***imax*** и ***imin***, используемых для хранения номеров максимального и минимального элемента, присваивается значение 1. Таким образом, за максимум и минимум принимается 1-й элемент массива, само же значение этого элемента нигде дополнительно не сохраняется. Оставшиеся элементы массива также перебираются, начиная со второго (блок 2). При этом каждый элемент массива X_i сравнивается с элементами, имеющими номер равный ***imax*** (блок 3) и ***imin*** (блок 6), т.е. с элементами принятыми за максимум (минимум). Если результат проверки условий является истиной, то в переменных ***imax*** и ***imin*** сохраняются индексы новых максимального (блок 4) и минимального (блок 6) элементов. После выхода из цикла будут выведены значения максимального и минимального элементов, а также их номера в массиве (блок 7).

Если в массиве имеется несколько элементов равных по значению максимальному (минимальному) элементу, то алгоритмы, приведенные на рис 5.2.1 и 5.2.2, определят позицию только первого такого элемента. Например, чтобы найти количество элементов массива равных максимальному и позицию последнего такого элемента, можно воспользоваться алгоритмом, приведенным на рис. 5.2.3.

В начале алгоритма за максимум принимается 1-й элемент массива и вводится переменная ***k*** для подсчета количества элементов равных максимальному (блок 1). Затем организовывается цикл по перебору оставшихся элементов массива (блок 2). На каждом шаге цикла выполняется следующая последовательность действий.

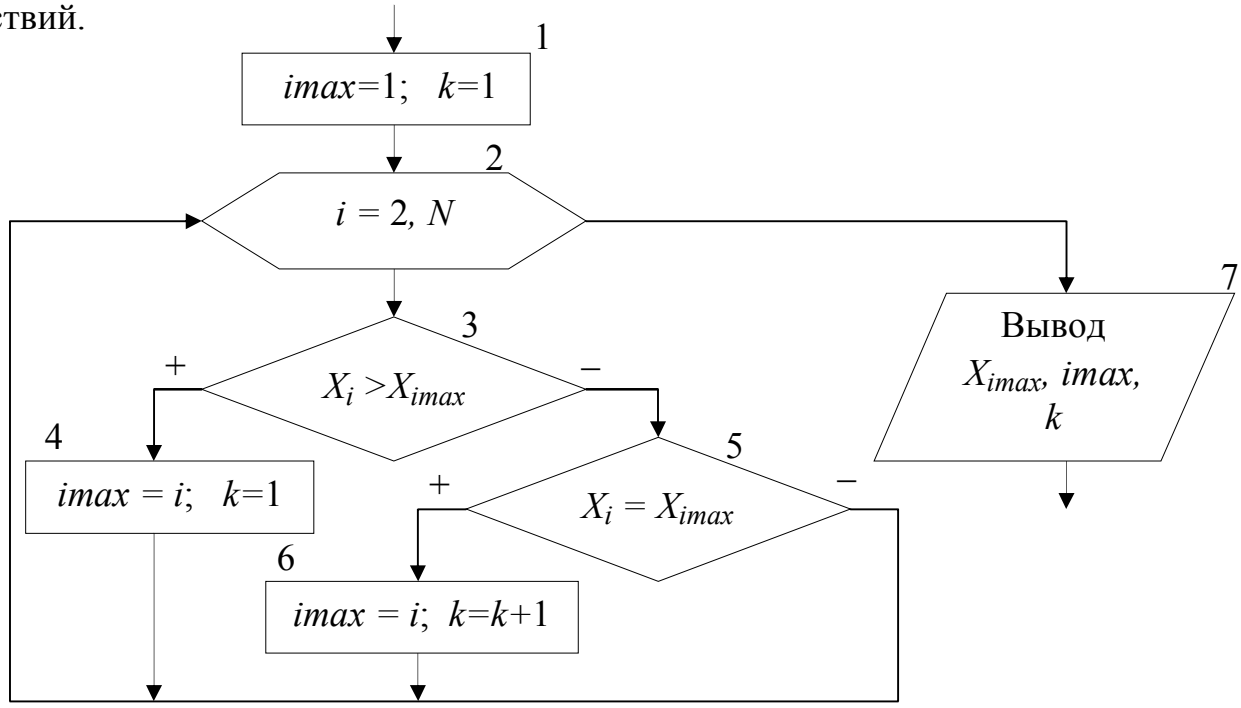


Рисунок 5.2.3. Определение количества элементов массива равных максимуму

Вначале текущий элемент массива X_i сравнивается с максимальным (блок 3). Если он оказывается больше элемента, принятого за максимальный, то запоминается номер этого элемента и переменной ***k*** присваивается 1 (блок 4). Это означает, что встретился первый «новый» максимальный элемент.

Если текущий элемент не больше максимума, значит, он может быть равным или меньше максимального элемента. Поэтому в блоке 5 текущий элемент массива X_i проверяется на равенство максимальному. Если он оказывается равным максимуму, то опять запоминается номер текущего элемента и значение переменной ***k*** увеличивается на 1 (блок 6). Это означает, что встретился следующий элемент равный максимальному элементу. В итоге после выхода из цикла в переменной ***imax*** будет храниться индекс последнего по счету элемента равного максимальному, а в переменной ***k*** количество элементов, равных максимуму. Если же из блока 6 убрать операцию ***imax=i***, то в переменной ***imax***, как и в предыдущих примерах будет сохранен номер первого по счету элемента равного максимальному.

В некоторых задачах невозможно вначале принять за максимум (минимум)

первый элемент массива. Например, если элементы массива ещё не определены (не введены или не вычислены), или если необходимо найти максимум (минимум) только среди положительных, четных по значению и т.д. элементов. Во втором случае неизвестно когда в массиве встретится первый такой элемент. В таких ситуациях можно использовать способ, реализованный в алгоритме из примера 5.2.

Пример 5.2. В массиве X [N] найти минимальный положительный элемент.

Массив X может содержать как положительные, так и отрицательные элементы. Поэтому неизвестно какой элемент массива или произвольное значение можно принять за начальный минимум. Решение поставленной задачи можно разбить на два этапа: поиск положительных элементов в массиве с определением первого такого элемента и поиск среди положительных элементов минимального по значению. Блок-схема алгоритма приведена на рис. 5.2.4.

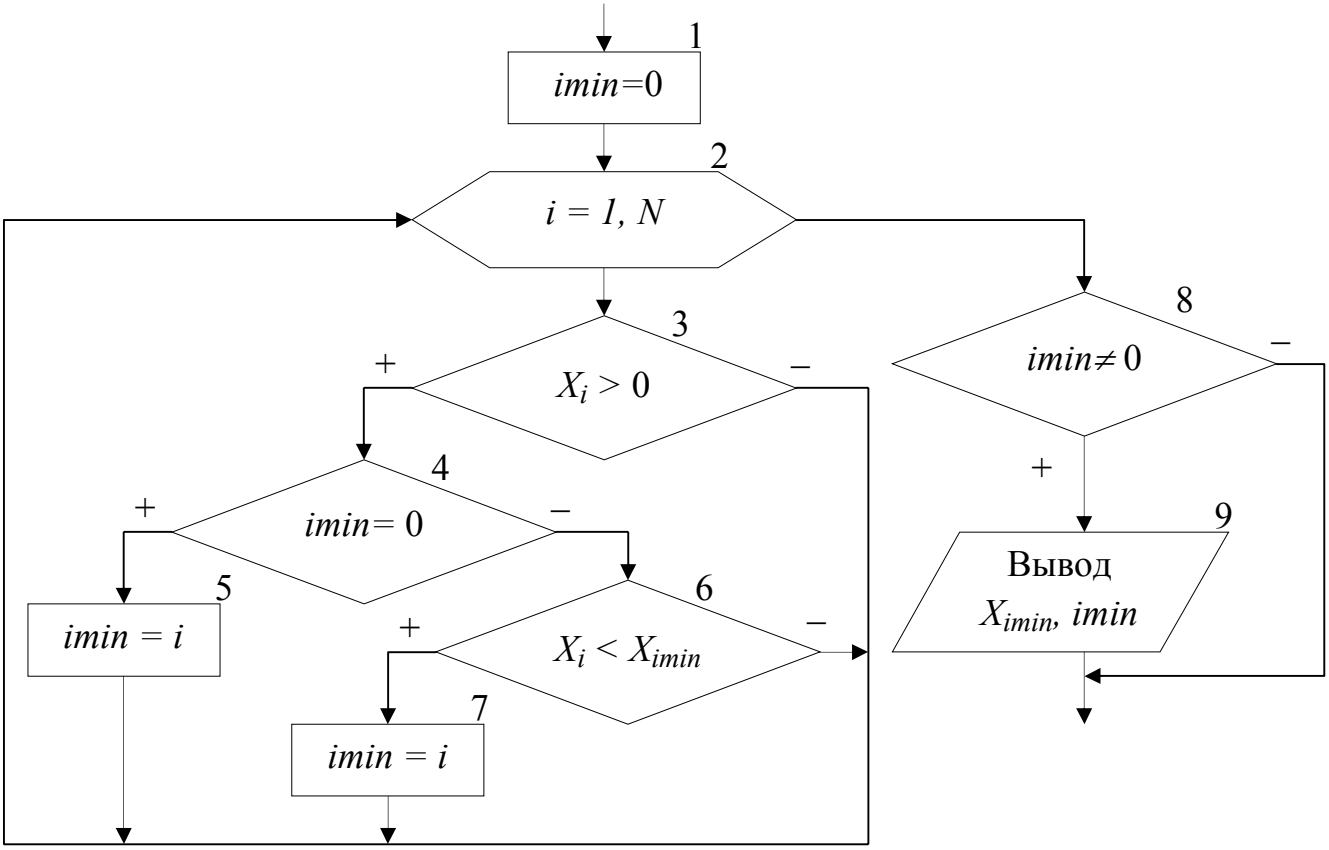


Рисунок 5.2.4. Блок-схема алгоритма для примера 5.2.

В нашем примере нумерация элементов массива начинается с единицы, поэтому в качестве начального значения для переменной ***imin*** принимается 0 (блок 1), т.е. значение которое не могут принимать индексы элементов массива X. Таким образом показывается, что при обработке массива еще не встречался положительный элемент, который можно принять за начальное значение для минимума. Затем организовывается цикл по перебору всех элементов массива (блок 2). На

каждом шаге цикла выполняется следующая последовательность действий.

В блоке 3 проверяется, не является ли текущий элемент массива X_i положительным. Если он отрицательный или равен нулю, то такой элемент пропускается и над ним не выполняется никаких действий. Если $X_i > 0$, то в блоке 4 проверяется, не является ли этот элемент первым встретившимся положительным элементом массива (признаком чего служит значение ***imin*** равное 0). В этом случае в блоке 5 переменной ***imin*** будет присвоено текущее значение ***i***. Таким образом, за начальное значение для минимума будет принято значение первого по счету положительного элемента массива X. Эта ситуация может возникнуть только один раз, и при дальнейшей работе цикла блок 5 больше выполняться не будет. Остальные положительные элементы массива в блоке 6 будут сравниваться с элементом массива, принятым в текущий момент за минимум. Если такой элемент окажется меньше минимального, то в блоке 7 в переменной ***imin*** будет сохранен его номер ***i***.

После выхода из цикла проверяется, не равно ли значение ***imin*** нулю (блок 8), так как сразу же выводить значение минимального элемента массива X_{imin} и его номер ***imin*** (блок 9) нельзя. Это объясняется тем что, если в массиве X не будет положительных элементов, то значение переменной ***imin*** останется равным нулю, а обращение к несуществующему элементу массива X_0 не допустимо.

5.3. Сортировка элементов массива

Под сортировкой элементов массива понимается упорядочение элементов массива в порядке возрастания (убывания) их значений. Для этого необходимо организовать два вложенных цикла. Внешний цикл будет перебирать элементы, начиная с 1-го до предпоследнего, и выбирать так называемый «главный» элемент. Внутренний цикл будет перебирать элементы, начиная со следующего после текущего «главного» и до последнего.

Алгоритмы сортировки элементов массива по возрастанию или убыванию полностью идентичны и отличаются только проверяемым во внутреннем цикле условием. Поэтому в качестве примера рассмотрим алгоритм сортировки элементов массива по возрастанию их значений (рис. 5.3).

Внешний цикл «Для» на основе блока модификации (блок 1) по параметру ***i*** будет перебирать все элементы массива X, начиная с первого и заканчивая предпоследним (***N*-1**-ым) элементом. Для каждого выбранного во внешнем цикле

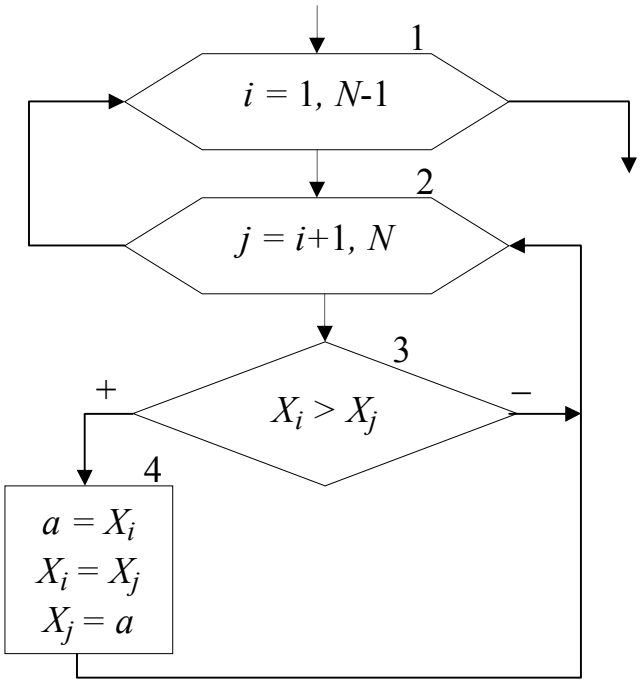


Рисунок 5.3. Сортировка элементов массива по возрастанию

«главного» элемента X_i организовывается внутренний цикл «Для» на основе блока модификации (блок 2), который будет перебирать элементы того же самого массива X, только по параметру ***j***. При этом перебираться будут элементы со следующего (***i***+1) после текущего «главного» элемента и заканчивая последним (***N***-ым) элементом. Таким образом, выбранный ***i***-й элемент во внутреннем цикле (блок 3) поочередно будет сравниваться со всеми стоящими после него в массиве элементами (элементами с индексом ***j***). Если «главный» ***i***-й элемент оказывается больше, чем последующий ***j***-й элемент, то их значения меняются местами (блок 4). Обмен значений двух элементов массива выполняется за 3 шага, с использованием дополнительной переменной ***a***.

В процессе этих обменов в ***i***-ю позицию будут выставляться все меньшие и меньшие значения. После завершения работы внутреннего цикла в ***i***-м элементе массива будет находиться наименьшее, среди находящихся после «главного» элемента, значение. Затем во внешнем цикле выбирается следующий «главный» элемент, для которого выполняются те же самые действия и т.д. В итоге весь массив будет отсортирован в порядке возрастания значений его элементов.

Пусть массив X состоит из четырех элементов {5; 7; 3; 2}. Пошаговое выполнение внутреннего цикла, при выбранном в качестве «главного» 1-го, а затем 2-го элементов массива, приведено ниже:

№ шага	X_1	X_2	X_3	X_4
1	**5**	**7**	3	2
	↑***i*** =1	↑***j***=**2**		
2	**5** ◄	7	► **3**	2
	↑***i*** =1		↑***j***=**3**	
3	**3** ◄	7	5	► **2**
	↑***i*** =1			↑ ***j***=**4**
Итог	**2**	7	5	3

№ шага	X_1	X_2	X_3	X_4
1	2	**7** ◄► **5**		3
		↑***i*** =**2**	↑***j***=**3**	
2	2	**5** ◄	7	► **3**
		↑***i*** =**2**		↑***j***=**4**
Итог	**2**	**3**	7	5

В итоге вначале в 1-й элемент массива занесено наименьшее значение. Затем во 2-й элемент массива занесено наименьшее среди оставшихся значение и т.д.

Для того чтобы этот алгоритм выполнял сортировку по убыванию достаточно в блоке 3 поменять знак больше на знак меньше.

5.4. Циклический сдвиг элементов массива

Под циклическим сдвигом понимается перестановка значений всех элементов массива на одну или несколько позиций влево или вправо. При этом значение первого (последнего) элемента массива заносится в последний (первый) элемент массива в зависимости от направления сдвига.

Схематичное изображение циклического сдвига элементов массива вправо на одну позицию, выполняемого в три этапа, показано на рис. 5.4.1. Блок-схема алгоритма, выполняющего такой сдвиг, показана на рис. 5.4.2.

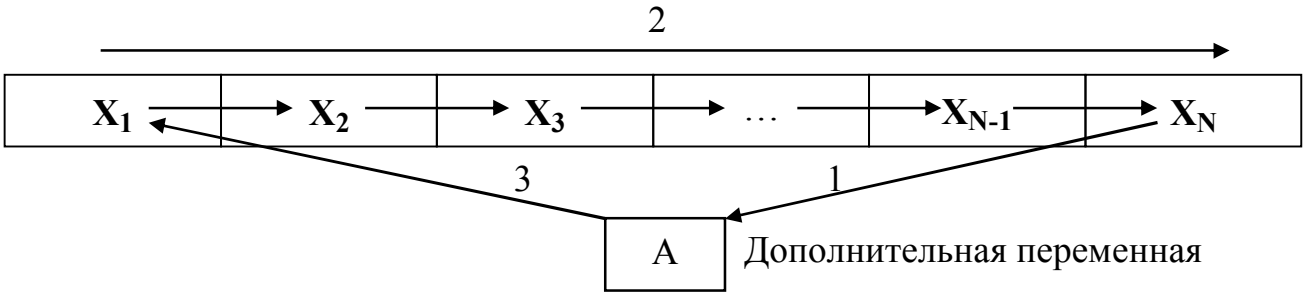


Рисунок 5.4.1. Принцип циклического сдвига элементов массива вправо

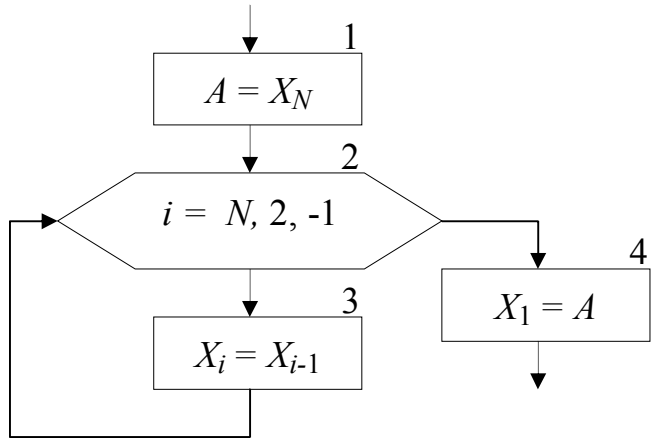


Рисунок 5.4.2. Циклический сдвиг вправо элементов массива

На 1-м этапе значение последнего элемента массива X_N заносится в дополнительную переменную A (блок 1). На 2-м этапе организовывается цикл «Для» на основе блока модификации (блок 2), который перебирает элементы массива X в обратном порядке, т.е. с правого края (шаг -1), начиная с N-го и заканчивая 2-м элементом. На каждом шаге цикла текущему элементу X_i присваивается значение предыдущего элемента X_{i-1} (блок 3). В результате выполнения 2-го этапа значение предва-

рительно сохраненного последнего элемента массива будет удалено. Значения остальных элементов массива будут смещены на одну позицию вправо. Значение первого элемента массива будет продублировано во втором элементе. На 3-м этапе алгоритма значение дополнительной переменной ***A***, в которой хранится последний элемент исходного массива, будет занесено в первый элемент массива X_1. В итоге будет выполнен циклический сдвиг элементов массива X на одну позицию вправо.

Следует обратить внимание на граничные значения параметра цикла ***i***, который изменяется от ***N*** до 2. При подстановке этих значений в формулу сдвига не должно происходить обращение к несуществующим элементам массива. При $i = N$ формула сдвига принимает вид $X_N = X_{N-1}$, при $i = 2$ – $X_2 = X_1$. В случае если бы правая граница параметра ***i*** была равна 1, то в формуле сдвига происходило бы обращение к несуществующему элементу массива с индексом 0: $X_1 = X_0$, что недопустимо.

Схематичное изображение циклического сдвига элементов массива влево на одну позицию, также выполняемого в три этапа, показано на рис. 5.4.3. Блок-схема алгоритма, выполняющего такой сдвиг, показана на рис. 5.4.4.

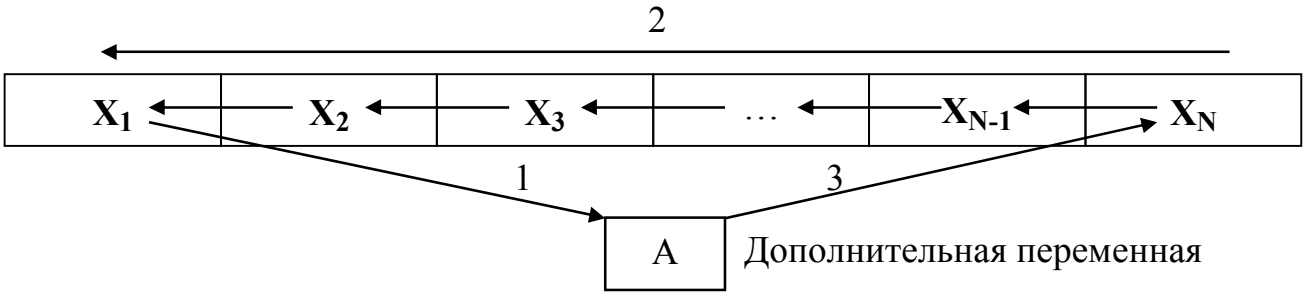


Рисунок 5.4.3. Принцип циклического сдвига элементов массива влево

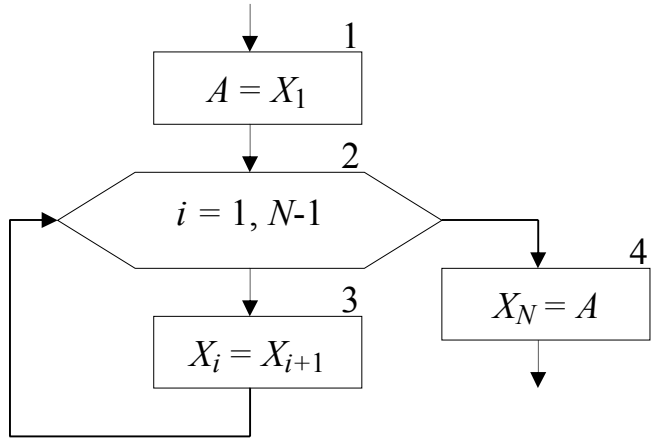


Рисунок 5.4.4. Циклический сдвиг влево элементов массива

На 1-м этапе значение первого элемента массива X_1 заносится в дополнительную переменную ***A*** (блок 1). На 2-м этапе организовывается цикл «Для» (блок 2), который перебирает элементы массива X в прямом порядке, т.е. с левого края (шаг +1), начиная с 1-го и заканчивая ***N***-1-м элементом. На каждом шаге цикла текущему элементу

X_i присваивается значение последующего элемента массива X_{i+1} (блок 3). В результате выполнения 2-го этапа значение предварительно сохраненного первого элемента массива будет удалено. Значения остальных элементов массива будут смещены на одну позицию влево. Значение последнего элемента массива будет продублировано в предпоследнем элементе. На 3-м этапе алгоритма значение дополнительной переменной A, в которой хранится первый элемент исходного массива, будет занесено в последний элемент массива X_N. В итоге будет выполнен циклический сдвиг элементов массива X на одну позицию влево.

Для выполнения циклического сдвига элементов массива на k позиций достаточно k раз выполнить алгоритм сдвига на одну позицию, т.е. добавить внешний цикл, который будет требуемое количество раз повторять сдвиг на одну позицию. Пример блок-схемы алгоритма, выполняющего циклический сдвиг элементов массива на k позиций влево, представлен на рис. 5.4.5.

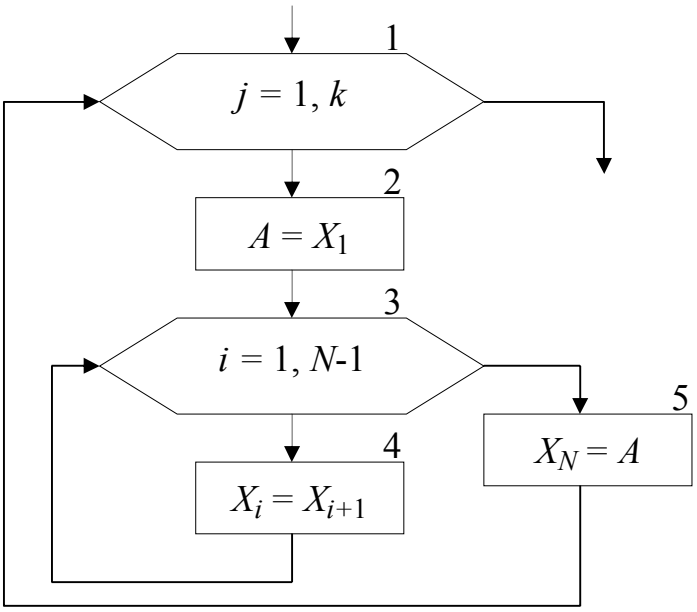


Рисунок 5.4.5. Циклический сдвиг влево на k позиций

Сдвиг элементов массива находит применение при удалении или добавлении элементов в массив, а также в других задачах.

5.5. Добавление и удаление элементов массива

Для добавления нового элемента массива необходимо выполнить следующую последовательность действий: увеличить размерность на единицу; сдвинуть вправо на одну позицию часть массива, начиная с позиции, в которую добавляется элемент; в «освободившийся» элемент записать добавляемое значение.

Пример 5.5.1. В массив X размерностью N добавить новый элемент. Значение нового элемента и позиция, в которую он добавляется, указываются при вводе.

На рис. 5.5.1. приведен фрагмент блок-схемы алгоритма решения поставленной задачи. Процесс ввода и вывода массива X не показан, т.к. он не отличается от описанного в п. 5.1.

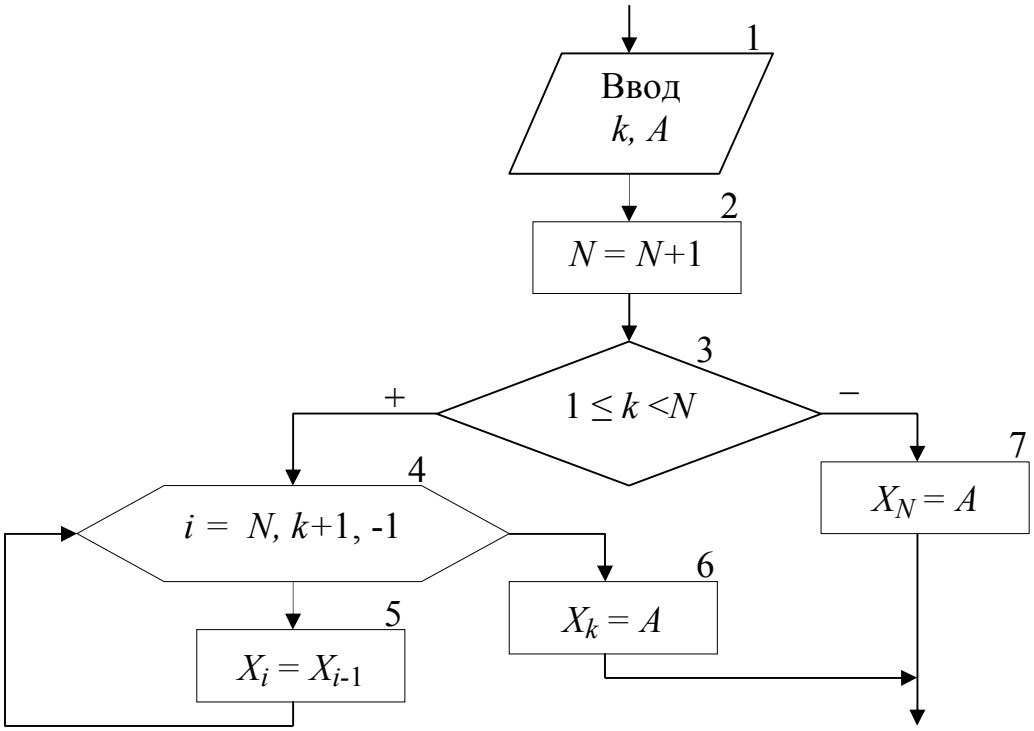


Рисунок 5.5.1. Добавление нового элемента массива в k-ю позицию

В блоке 1 в переменную ***A*** вводится значение нового элемента массива, а в переменную ***k*** заносится номер позиции, в которую он будет добавляться. После этого увеличивается размерность массива ***N*** на единицу (блок 2) и проверяется, действительно ли новый элемент добавляется в середину массива (блок 3). В этом случае производится сдвиг части элементов массива X с ***N***-й до ***k***-й позиции вправо (блоки 4-5) и значение переменной ***A*** заносится в «освободившийся» элемент массива с номером ***k*** (блок 6). Если условие в блоке 3 не выполняется, то новый элемент добавляется в конец массива X (блок 7).

Для удаления элемента из массива достаточно выполнить сдвиг части массива, расположенного после удаляемого, на одну позицию влево. В результате будет стерт удаляемый элемент, а последний элемент массива будет продублирован в предпоследний. После этого достаточно уменьшить размерность массива на единицу.

Пример 5.5.2. Из массива X размерностью ***N*** удалить элемент с номером ***k***. Номер удаляемого элемента указывается при вводе.

На рис. 5.5.2. показан фрагмент блок-схемы алгоритма решения поставленной задачи. При этом процесс ввода и вывода массива X также не приводится.

В блоке 1 в переменную ***k*** вводится номер элемента, который необходимо удалить. В блоке 2 проверяется, находится ли удаляемый элемент в середине массива. В этом случае производится сдвиг части элементов массива X с ***k***-й до ***N***-й

позиции влево (блоки 3-4) и уменьшается размерность массива ***N*** на единицу (блок 7). В противном случае проверяется, не является ли удаляемый элемент последним в массиве (блок 5). Если это так, то для удаления элемента достаточно всего лишь уменьшить размерность массива ***N*** (блок 7). Если же значение k не удовлетворяет этим двум условием, значит, оно задано не верно и удаление элемента из массива не производится (блок 6).

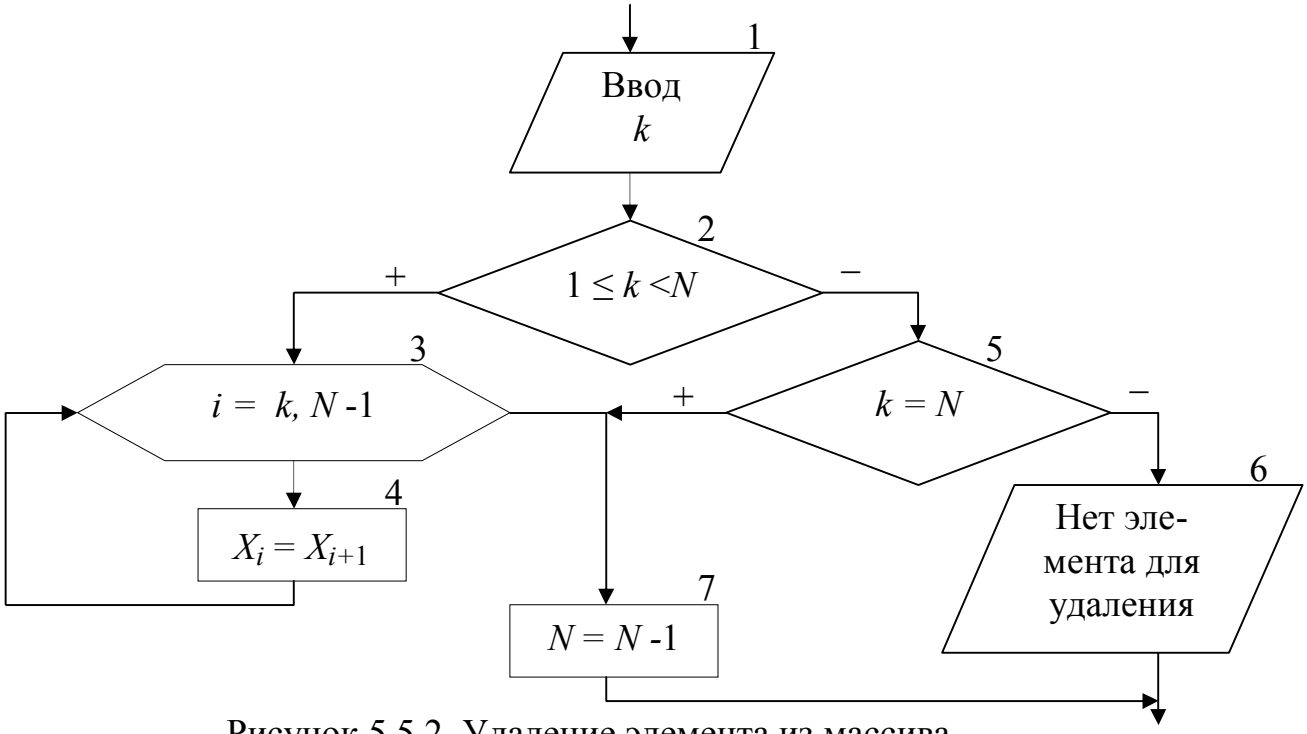


Рисунок 5.5.2. Удаление элемента из массива

Иногда возникает необходимость удалить из массива не один, а несколько элементов, удовлетворяющих определенному условию (например, все элементы, значение которых равно 0). Для этой цели необходимо организовать два цикла по перебору элементов массива. Первый цикл (внешний) будет искать в массиве элементы, которые необходимо удалить. Второй цикл (внутренний) будет удалять найденные элементы, описанным выше способом. При этом для организации внешнего цикла не рекомендуется использовать цикл «Для». Это связано с тем, что после удаления элемента во внутреннем цикле, будет уменьшаться размерность массива, которая в свою очередь является граничным значением для параметра внешнего цикла.

На рис. 5.5.3. приводится фрагмент блок-схемы алгоритма, в котором выполняется удаление всех нулевых элементов из массива за один цикл.

Переменная ***k*** выполняет две функции: является счетчиком элементов массива, которые не надо удалять (не равные 0), и в тоже время задает позицию в массиве X, в которую переставляется такой элемент (блок 1). В массиве осущест-

вляется поиск (блоки 2-3) и подсчет количества элементов, которые не должны быть удалены (блок 4); каждый найденный элемент переставляется в начало массива в позицию, совпадающую с порядковым номером этого элемента (блок 4).

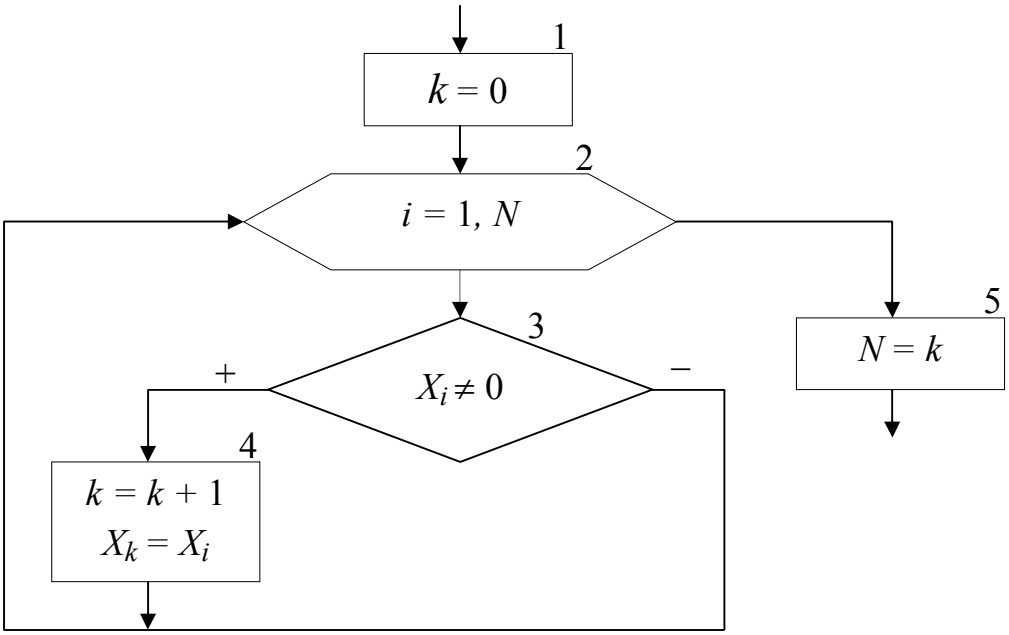


Рисунок 5.5.3. Удаление из массива нулевых элементов

В результате все нулевые элементы массива будут пропущены, а элементы не равные 0 выставлены подряд в начало массива, размерность которого уже будет равна ***k*** (блок 5).

6. АЛГОРИТМЫ ОБРАБОТКИ ДВУМЕРНЫХ МАССИВОВ

Двумерный массив (матрица) представляет собой таблицу, на пересечении строк и столбцов которой располагаются элементы. Каждый элемент имеет два индекса. Первый индекс обычно обозначается буквой ***i*** и указывает номер строки, в которой расположен элемент. Второй индекс обозначается буквой ***j*** и указывает номер столбца, в котором расположен элемент (рис 6.1). Размерность двумерного массива задается двумя числами: ***M*** – количество строк и ***N*** – количество столбцов.

	j=1	j=2	j=3	...	j=N
i=1	$A_{1,1}$	$A_{1,2}$	$A_{1,2}$	···	$A_{1,N}$
i=2	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	···	$A_{2,N}$
...	···	...	...	$A_{i,j}$	...
i=M	$A_{M,1}$	$A_{M,2}$	$A_{M,3}$	···	$A_{M,N}$

Рисунок 6.1. Двумерный массив

Двумерный массив, у которого количество строк равно количеству столбцов называется квадратной матрицей, в противном случае – прямоугольной.

Для обработки двумерного массива требуется два вложенных цикла, при этом наиболее удобно использовать циклы «Для» на основе блока модификации. Один будет перебирать строки, второй – столбцы массива. Таким образом, будут перебраны все элементы массива.

Ввод двумерного массива (рис. 6.2), также как и одномерного выполняется в два этапа. Вначале вводится размерность массива (блок 1), а затем значения для каждого элемента. Внешний цикл (блок 2) при ***i*** =1 «выбирает» 1-ю строку массива. Внутренний цикл (блок 3) перебирает все столбцы массива, т.е. поочередно выбираются элементы $A_{1,1}$, $A_{1,2}$, $A_{1,3}$ и т.д. до конца 1-й строки и вводятся их значения (блок 4). После выхода из внутреннего цикла происходит возврат в блок 2, где выбирается 2-я строка массива, для которой внутренний цикл опять переберет поочередно все элементы $A_{2,1}$, $A_{2,2}$, $A_{2,3}$ и т.д. Таким образом, элементы двумерного массива будут перебираться по строкам.

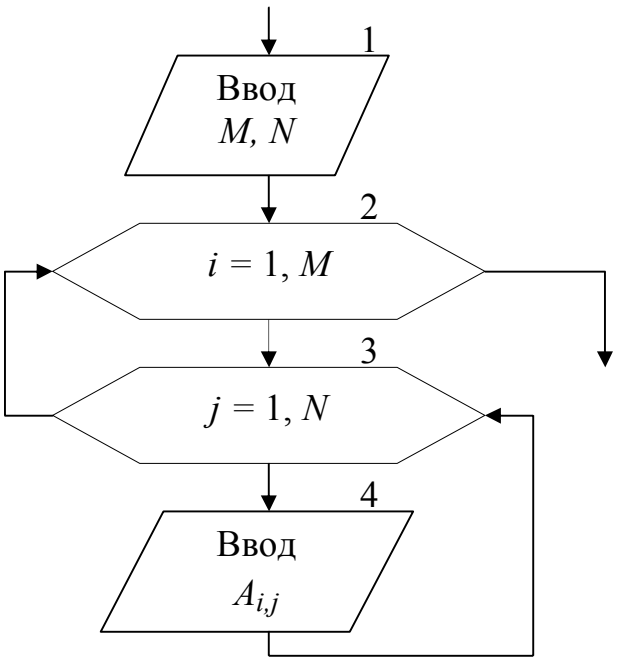


Рисунок 6.2. Ввод двумерного массива

Аналогичным образом выполняется вывод элементов двумерного массива.

Если в блок-схеме на рис. 6.2. поменять местами параметры внешнего и внутреннего циклов, т.е. внешний цикл сделать по параметру ***j***, а внутренний – по параметру ***i***, то элементы массива будут перебираться по столбцам.

Пример 6.1. Сформировать вектор B размерностью M, каждый элемент которого равен количеству нулевых элементов соответствующей строки матрицы A размерностью M на N.

Как видно из условия количество элементов вектора B равно количеству строк матрицы A. Для решения поставленной задачи необходимо организовать построчный перебор элементов двумерного массива. Внешним должен быть цикл по параметру ***i***, внутренним цикл по параметру ***j***. Это даст возможность после завер-

шения обработки каждой строки формировать элементы одномерного массива B. Блок-схема алгоритма приведена на рис. 6.3.

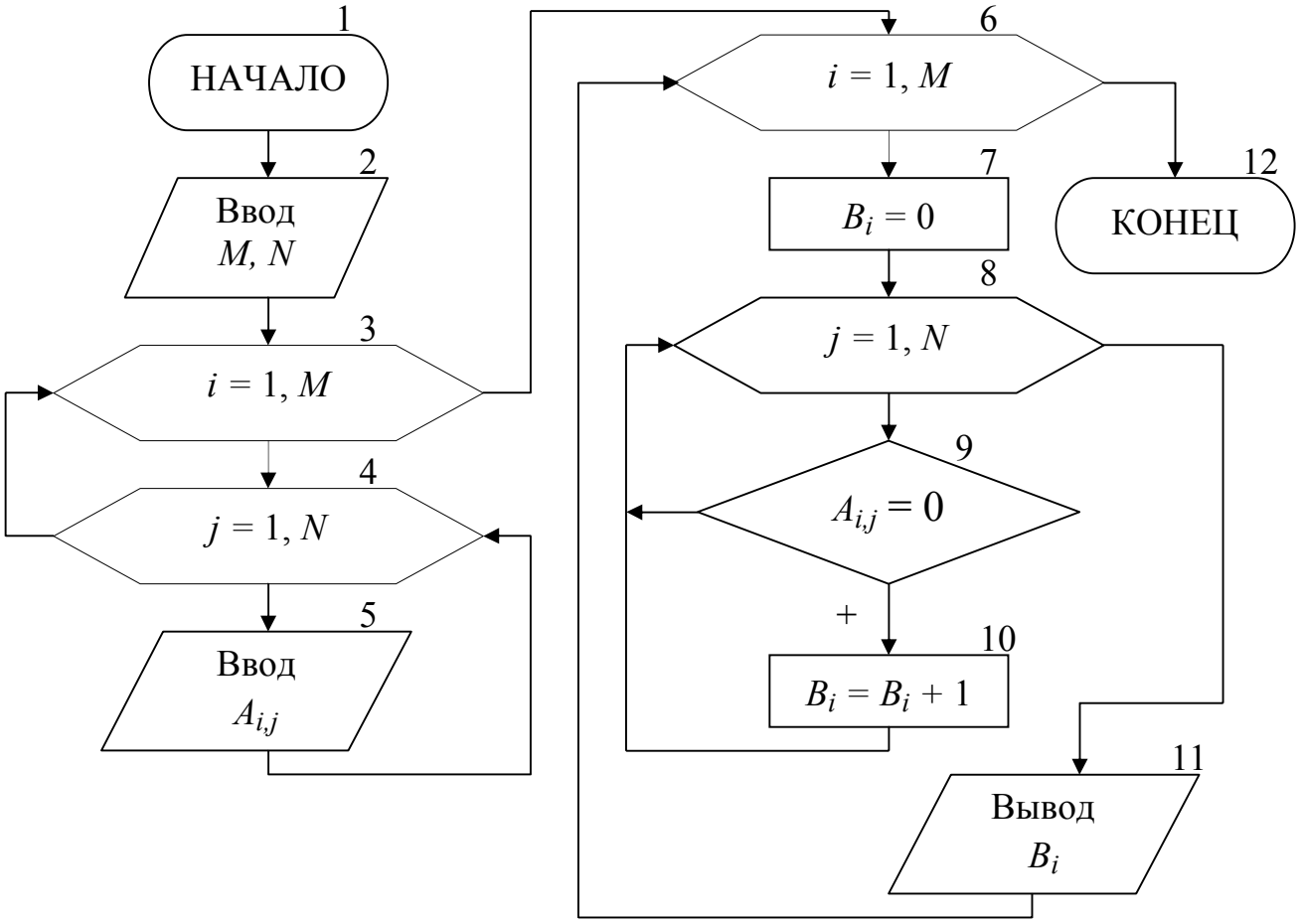


Рисунок 6.3. Блок-схема алгоритма для примера 6.1.

Блоки 2-5 вводят исходный двумерный массив A, описанным выше способом. Затем организовывается внешний цикл «Для» по параметру ***i*** на основе блока модификации (блок 6), который будет одновременно перебирать строки массива A и элементы одномерного массива B. С целью оптимизации алгоритма не вводится отдельная переменная для хранения количества нулевых элементов в каждой строке матрицы. Вместо неё будут использоваться непосредственно элементы массива B.

Для выбранной во внешнем цикле ***i***-й строки обнуляется ***i***-й элемент массива B (блок 7). Затем организовывается внутренний цикл «Для» по параметру ***j*** (блок 8), перебирающий столбцы массива A, т.е. элементы ***i***-й строки. Каждый элемент проверяется на равенство нулю (блок 9), и в случае выполнения условия происходит увеличение счетчика нулевых элементов ***i***-й строки, значение которого хранится в элементе $\boldsymbol{B_i}$ (блок 10). После завершения обработки строки (выхода из внутреннего цикла) происходит вывод ***i***-го элемента массива B (блок 11) и пе-

реход к следующей строке. Обработав все строки двумерного массива A, алгоритм завершит свою работу.

Пример 6.2. Сформировать вектор B размерностью N, каждый элемент которого равен среднему арифметическому значению элементов соответствующего столбца матрицы A размерностью M на N.

В данном примере количество элементов вектора B равно количеству столбцов матрицы A. Для решения задачи необходимо организовать перебор элементов двумерного массива по столбцам. Внешним должен быть цикл по параметру ***j***, внутренним цикл по параметру ***i***. Это даст возможность после завершения обработки каждого столбца вычислять соответствующие элементы одномерного массива B. Блок-схема алгоритма приведена на рис. 6.4.

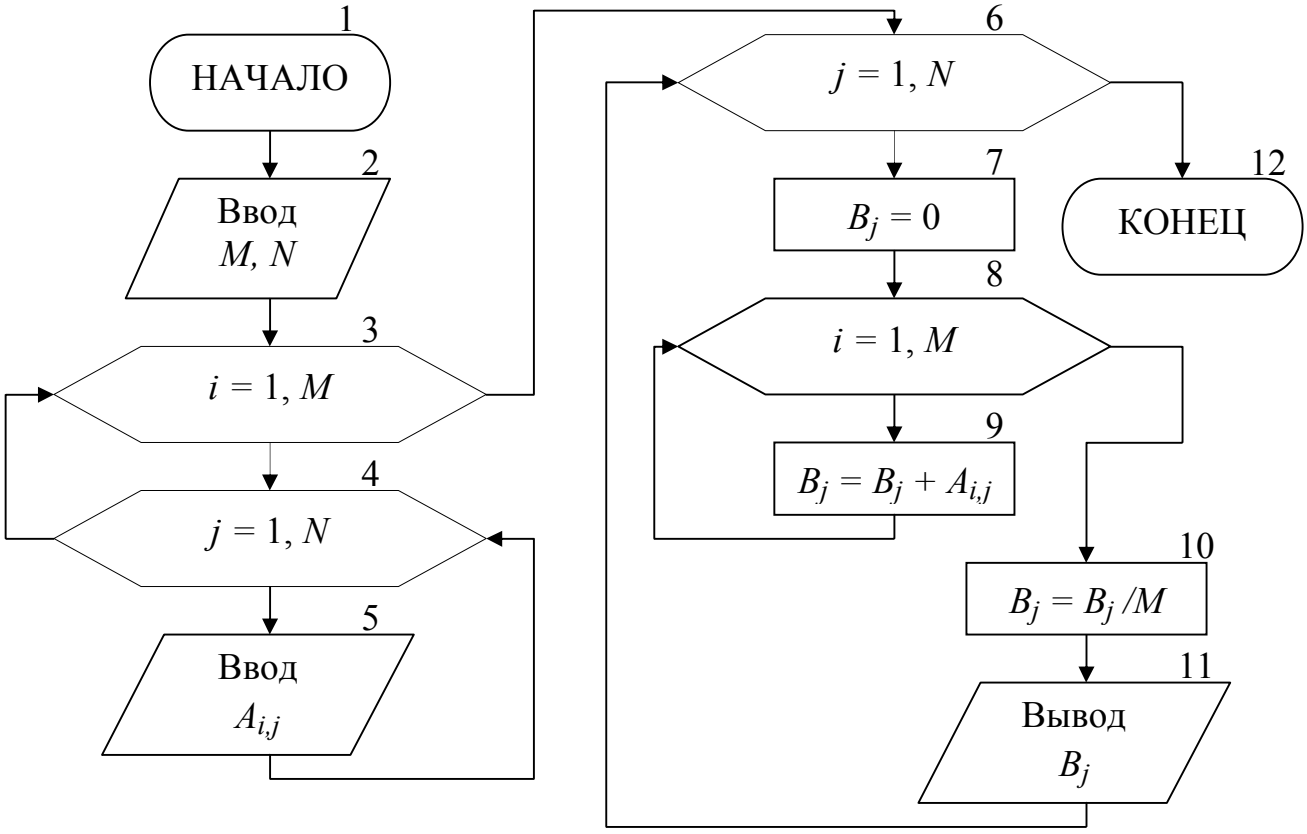


Рисунок 6.4. Блок-схема алгоритма для примера 6.2.

Ввод элементов массива A осуществляется построчно (блоки 2-5). Во внешнем цикле по параметру ***j*** выбирается столбец массива A (блок 6), для него обнуляется значение суммы, которая будет хранится в соответствующем ***j***-м элементе массива B (блок 7). Внутренний цикл по параметру ***i*** (блок 8) выполняет перебор и суммирование всех элементов текущего ***j***-го столбца массива A (блок 9). После завершения работы внутреннего цикла в ***j***-м элементе массива B вычисляется среднее арифметическое значение элементов ***j***-го столбца массива A (блок 10),

которое затем выводится (блок 11). На этом заканчивается тело внешнего цикла и происходит переход на его начало, где выбирается следующий столбец матрицы. Обработав все столбцы двумерного массива A, алгоритм завершит свою работу.

Принципы поиска максимального или минимального элементов в двумерных массивах ничем не отличаются от аналогичных принципов, используемых для одномерных массивов. Только в качестве параметров такого элемента определяются номера строки и столбца.

Пример 6.3. В каждой строке квадратной матрицы A размерностью N на N найти наибольший элемент и поменять его местами с элементом главной диагонали.

Главной диагональю квадратной матрицы называется диагональ, соединяющая верхний левый угол матрицы с правым нижним углом. Для элементов, расположенных на главной диагонали соблюдается соотношение между индексами: $i=j$. Для элементов расположенных ниже главной диагонали: $i > j$. Для элементов расположенных выше главной диагонали: $i < j$. Перебирая элементы матрицы и проверяя соотношения между индексами, всегда можно определить, где расположен текущий элемент.

В данной задаче необходимо организовать построчный перебор элементов матрицы. В каждой строке будем искать максимальный элемент, при этом для однозначного его определения достаточно хранить только номер столбца, в котором он находится в текущей строке. Блок-схема алгоритма приведена на рис. 6.5.

В блоках 2-5 вводится построчно квадратная матрица A размерностью N на N. Внешний цикл по параметру i (блок 6) выбирает текущую i-ю строку матрицы, в которой будет выполняться поиск максимального элемента. Вначале за максимум принимается первый элемент i-й строки (блок 7). Внутренний цикл перебирает элементы строки, начиная со второго (блок 8), и сравнивает их с элементом принятым за максимальный (блок 9). Если текущий элемент оказывается больше, чем принятый за максимум, то в переменной $jmax$ запоминается его позиция в i-й строке, т.е. номер столбца, в котором этот элемент находится (блок 10). В противном случае текущий элемент пропускается. После выхода из внутреннего цикла в элемент главной диагонали $A_{i,i}$ текущей i-й строки заносится значение максимального элемента этой строки $A_{i,jmax}$ (блок 11). На этом заканчивается обработка текущей строки массива A (тело внешнего цикла) и происходит выбор следующей строки массива. После завершения обработки всех строк двумерного массива A, в

блоках 12-14 выводится преобразованная матрица и алгоритм завершает свою работу.

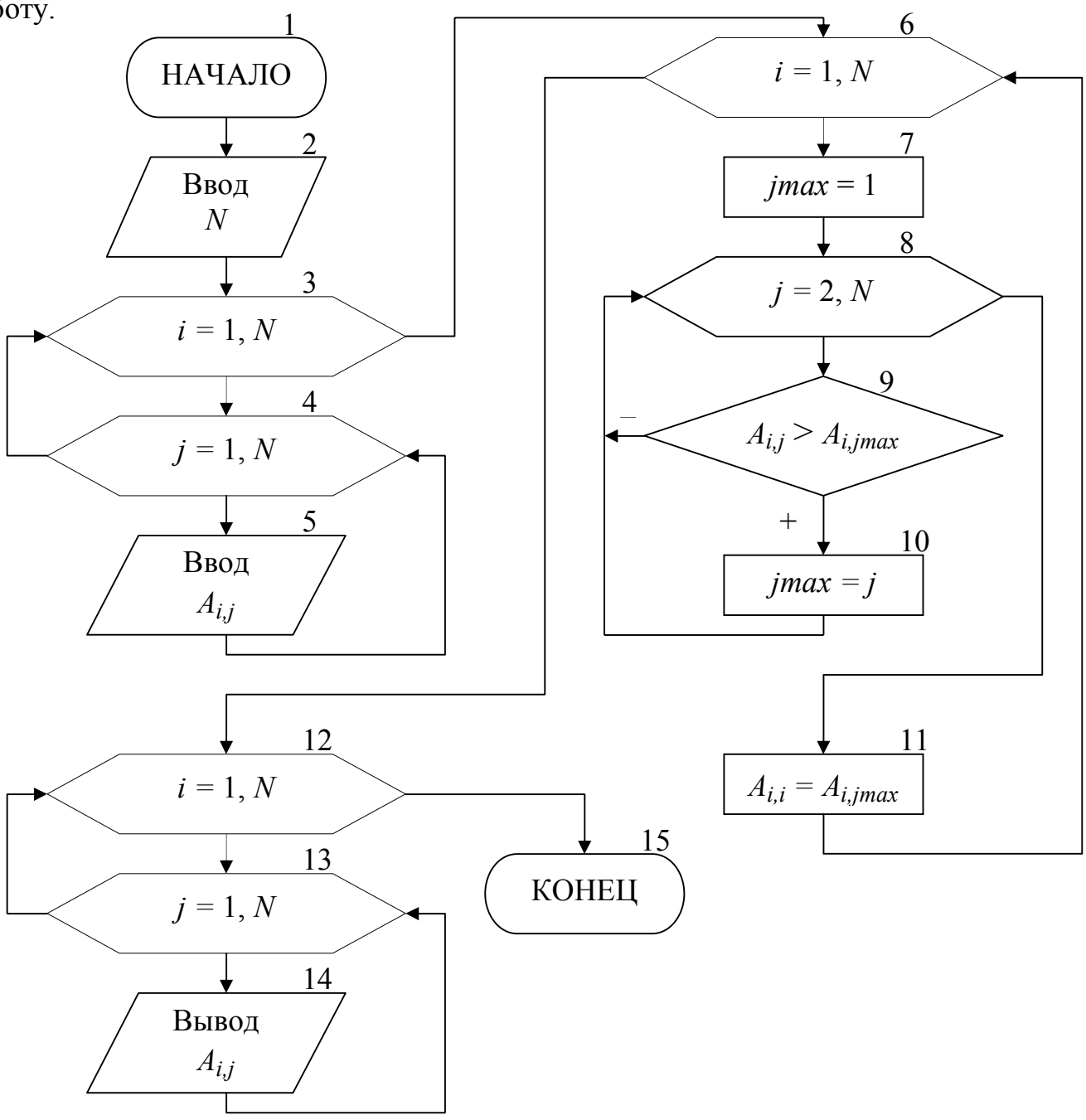


Рисунок 6.5. Блок-схема алгоритма для примера 6.3.

7. АЛГОРИТМЫ, СОДЕРЖАЩИЕ ВСПОМОГАТЕЛЬНЫЕ ПОДЗАДАЧИ

При решении многих расчетных задачах возникает необходимость повторить несколько раз одну и ту же последовательность действий в разных частях алгоритма. В этом случае рационально выделить повторяющуюся последователь-

ность действий в отдельную подзадачу – **подпрограмму**. Разработав алгоритм вспомогательной подзадачи, при составлении блок-схемы основного алгоритма необходимо предусмотреть обращение (вызов) к этой подзадаче.

Пример 7. Составить подпрограмму вычисления суммы элементов массива, которую использовать для обработки массивов A[Na], B[Nb]. Элементы массивов вычислять по формулам: $A_r=0.3\cdot r^2-1.2\cdot r$; $B_t=0.5\cdot t^2+1.7\cdot t$.

При решении поставленной задачи можно выделить несколько подзадач:

1. Подзадача вычисления (формирования) элементов исходных массивов A и B. Для этого следует обратить внимание на формульные зависимости по которым вычисляются элементы массивов и вывести общую зависимость. В нашем случае она будет иметь вид: $X_i = k1\cdot i^2+k2\cdot i$.

2. Подзадача вывода элементов исходных массивов.

3. Подзадача вычисления суммы элементов массивов.

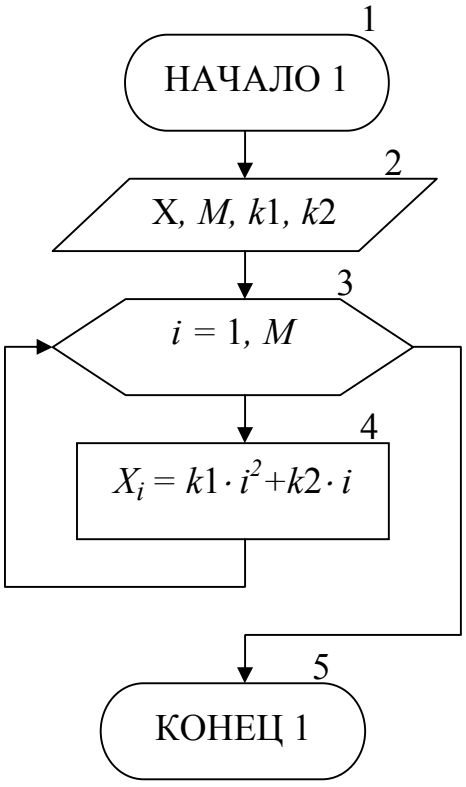


Рисунок 7.1. Подзадача расчета элементов массива pp1

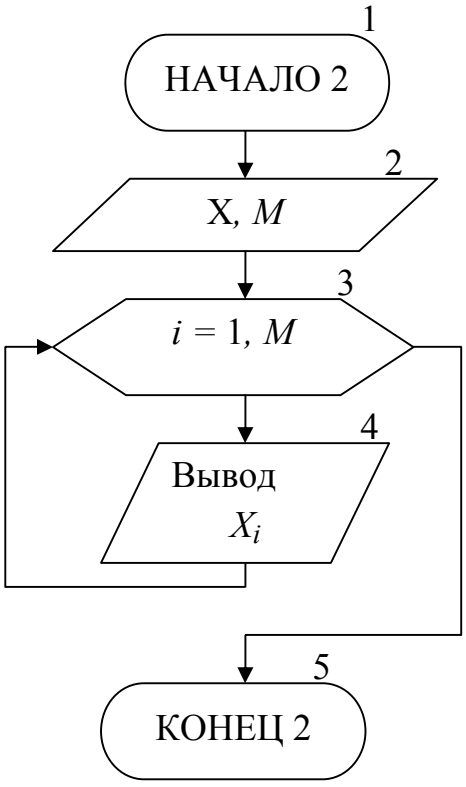


Рисунок 7.2. Подзадача вывода элементов массива pp2

На рис. 7.1 приведена блок-схема алгоритма подзадачи вычисления элементов внутреннего одномерного массива X. В качестве исходных данных в подзадачу передаются: массив X, размерность массива ***M***, коэффициенты ***k***1 и ***k***2.

На рис. 7.2 приведена блок-схема алгоритма подзадачи вывода элементов

внутреннего массива X. В качестве исходных данных в подзадачу передаются: массив X, размерность массива ***M***.

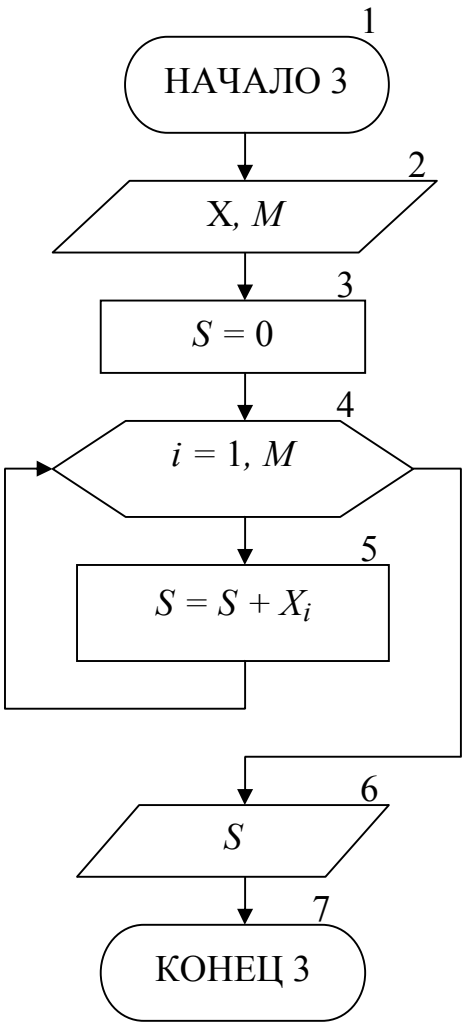


Рисунок 7.3. Подзадача вычисления суммы элементов массива pp3

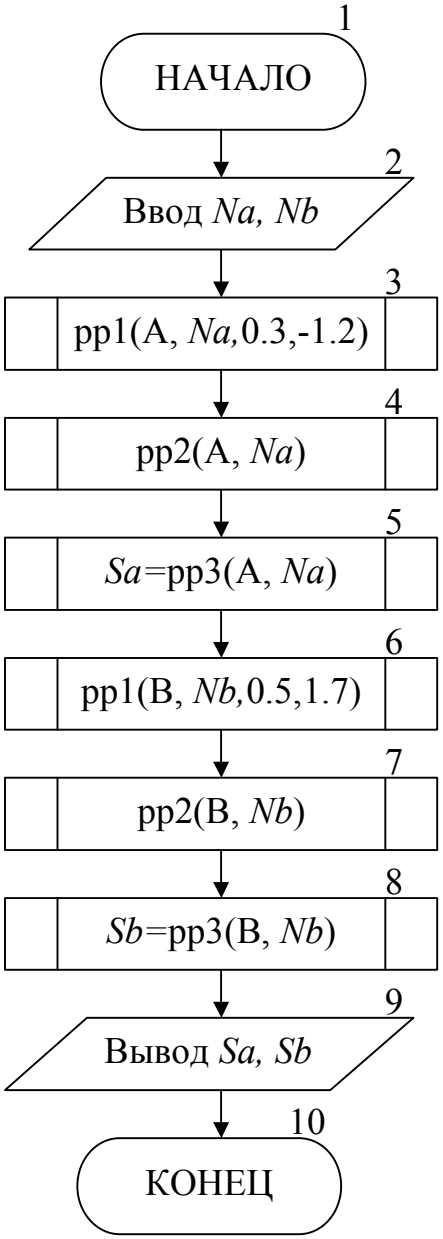


Рисунок 7.4. Основной алгоритм

На рис. 7.3 приведена блок-схема алгоритма подзадачи вычисления суммы элементов внутреннего массива X. В качестве исходных данных в подзадачу передаются: массив X, размерность массива ***M***. Результатом работы подзадачи будет вычисленное значение суммы элементов массива ***S***.

На рис. 7.4 приведена блок-схема основного алгоритма решения задачи. Вначале алгоритма вводятся размерности обрабатываемых массивов A и B (блок 2). Блок 3 выполняет обращение к подпрограмме расчета элементов массива pp1 и передает в неё фактические параметры массива A и коэффициенты для формульной зависимости. После завершения работы подпрограммы pp1 в основной алгоритм будет возвращен сформированный массив A. Затем в блоке 4 выполня-

ется вызов подпрограммы вывода элементов массива pp2, в которую также передаются фактические параметры массива A. Завершается обработка 1-го массива обращением к подпрограмме pp3 (блок 5), которая возвратит в переменную ***Sa*** значение вычисленной суммы элементов массива A.

Аналогичным образом в блоках 6-8 обрабатывается массив B. Завершает работу основного алгоритма вывод результирующих сумм ***Sa*** и ***Sb*** (блок 9).

Использование вспомогательных подзадач позволило избежать повторения одних и тех же по сути фрагментов блок-схемы алгоритма.

ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

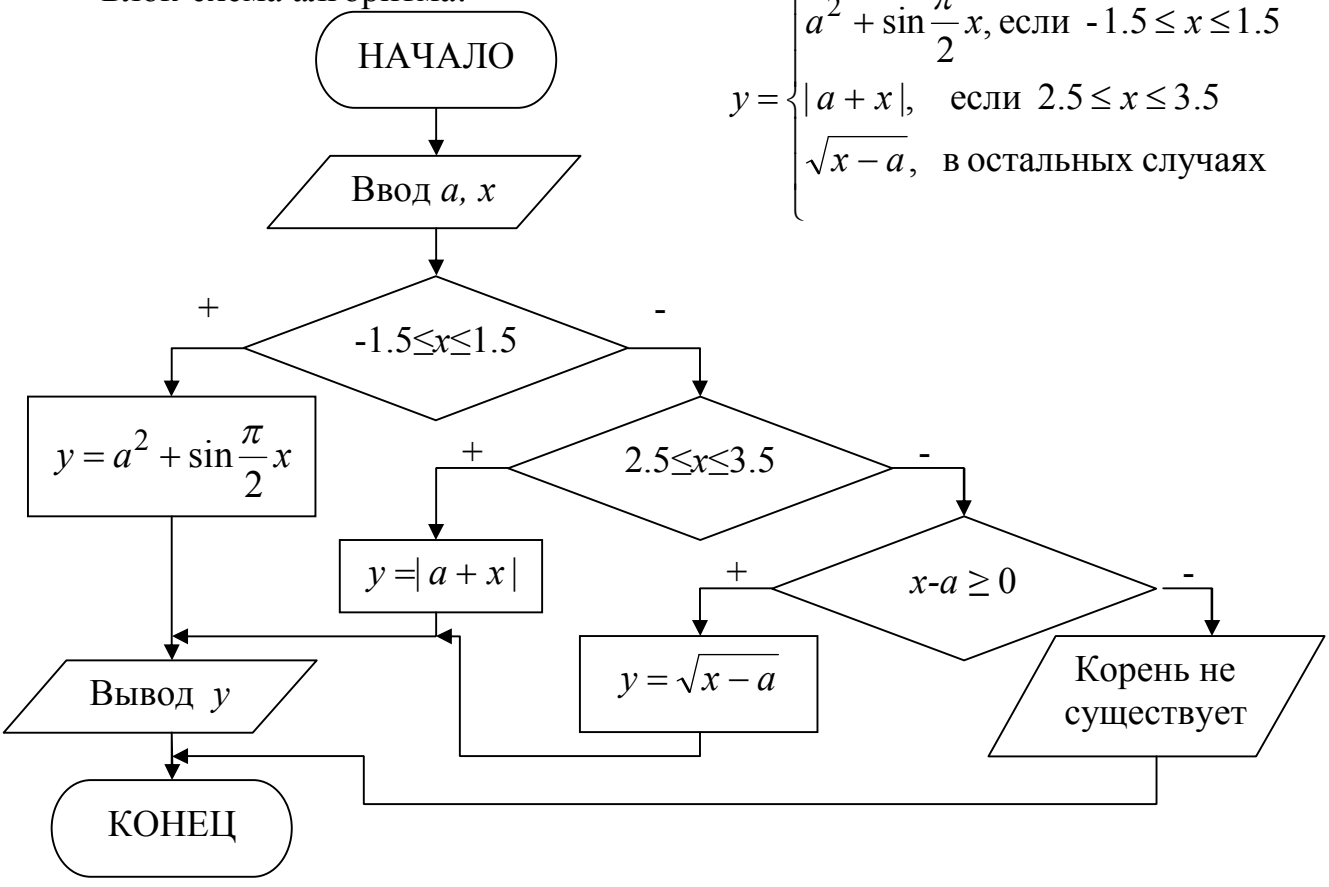
Задание №1.Организация линейного и разветвляющегося вычислительных процессов

Задание. Составить блок-схему алгоритма и программу решения поставленной задачи, которая в соответствии с исходными данными вычисляет значения заданных выражений.

Пример. Составить алгоритм и программу для вычисления значения ***y*** в соответствии с исходными данными ***x*** и ***a***.

Блок-схема алгоритма:

$$y=\begin{cases} a^2+\sin\dfrac{\pi}{2}x, \text{если } -1.5\le x\le 1.5 \\ |a+x|, \quad \text{если } 2.5\le x\le 3.5 \\ \sqrt{x-a}, \quad \text{в остальных случаях} \end{cases}$$



Программа на языке Турбо Паскаль:

```
Program Pr1;
Label m1;
Var
   a, x, y: real;
Begin
 Write('Ввод a и x');
 Readln(a,x);
 If (x >= -1.5) And (x <= 1.5) Then   y := Sqr(a)+ Sin(Pi / 2 * x)
 Else  If (x >= 2.5) And (x <= 3.5) Then  y := abs(a + x)
       Else  If x - a > = 0 Then y := Sqrt(x - a)
             Else  Begin  Writeln('Корень не существует');  Goto m1; End;
writeln('y=', y : 6 : 2);
m1:
End.
```

Варианты заданий.

№ п/п	Модель	Исходные данные	Выводимые данные
1	**2**	**3**	**4**
1	$y=\begin{cases} x-ab, & если\ x<4 \\ x+ab, & если\ 4\le x\le 5 \\ (x+a)/b, & если\ x>5 \end{cases}$ $x=\begin{cases} a+\mathrm{b}, & если\ a<b \\ a-\mathrm{b}, & если\ a\ge b \end{cases}$	a=3.5 b=4.7	*x, y.*
2	$y=\begin{cases} a+cx, & если\ x<1 \\ b+d/x, & если\ 1\le x<3 \\ c-ax, & если\ x\ge 3 \end{cases}$ $x=\begin{cases} \sqrt{ab}, & если\ ab\ge cd \\ \sqrt[3]{cd}, & если\ ab<cd \end{cases}$	a=2.5 b=1.3 c=1.5 d=2.3	*x, y.*
3	$z=\begin{cases} \ln ax, & если\ \lvert x\rvert<3 \\ bx^3, & если\ \lvert x\rvert=3 \\ cx-1, & если\ \lvert x\rvert>3 \end{cases}$ $x=\begin{cases} a+bc, & если\ ab\ge c \\ abc, & если\ ab<c \end{cases}$	a=1.2 b=2.5 c=3.1	*x, z.*

Продолжение таблицы к заданию 1

1	2	3	4
4	$y=\begin{cases}\sqrt[3]{a+x}, если\ x<1\\ \ln bx, \ если\ 1\le x\le 5\\ \sqrt{a+bx}, если\ x>5\end{cases}\quad x=\begin{cases}a^2b, \ если\ a<b\\ ab^2, \ если\ a\ge b\end{cases}$	a=1.5 b=2.1	x, y.
5	$z=\begin{cases}\sin^2 x+1, \ если\ x\le c\\ \cos x-1, \ если\ c<x<d\\ e^x+1/a, \ если\ x\ge d\end{cases}$ $x=\begin{cases}(a+c)d, \ если\ a<c\\ (a-c)/d, \ если\ a\ge c\end{cases}$	a=2.4 c=3.2 d=4.7	x, z.
6	$y=\begin{cases}a\sqrt[3]{x}, если\ x<1\\ bx, если\ 1\le x\le 3\\ cx^2, \ если\ x>3\end{cases}\ x=\begin{cases}ab+c, \ если\ a\le b+1\\ a/b-c, если\ a>b+1\end{cases}$	a=3.7 b=2.9 c=0.3	x, y.
7	$y=\begin{cases}x^3+1, \ если\ x<4\\ x^2+1, если\ 4\le x<5\\ x+1, \ если\ x\ge 5\end{cases}\ x=\begin{cases}a^2/b^2, если\ a\le b\\ a/b, \ если\ a>b\end{cases}$	a=1.3 b=4.5	x, y.
8	$y=\begin{cases}(a+b)x, если\ x<3\\ (a-b)x, если\ x=3\\ ax/b, \ если\ x>3\end{cases}\quad x=\begin{cases}a^2+1, \ если\ ab\ge 1\\ b^2-1, \ если\ ab<1\end{cases}$	a=3.6 b=2.3	x, y.
9	$y=\begin{cases}a+x, \ если\ x<5\\ ax, \ если\ 5\le x<7\\ x/a, \ если\ x\ge 7\end{cases}\quad x=\begin{cases}\sqrt{a^2+1}, \ если\ a\ge 2\\ \sqrt[3]{a^3+1}, \ если\ a<2\end{cases}$	a=2.5	x, y.
10	$y=\begin{cases}\sqrt{b+x^2}, если\ x<1\\ abx, если\ 1\le x\le 5\\ bx^3, \quad если\ x>5\end{cases}\ x=\begin{cases}ab+3, \quad если\ ab\le 3\\ a/b-3, \ если\ ab>3\end{cases}$	a=1.5 b=2.4	x, y.
11	$y=\begin{cases}a\sqrt{x}, если\ x<2\\ bx^2, если\ 2\le x<3\\ c\cdot e^x, \ если\ x\ge 3\end{cases}\ x=\begin{cases}(a+b)/c, если\ a\le b\\ (a-b)c, \ если\ a>b\end{cases}$	a=5.4 b=2.4 c=1.9	x, y.

Продолжение таблицы к заданию 1

1	2	3	4
12	$y=\begin{cases} b\cdot e^{x}, \ если\ x=2 \\ \sin x, \ если\ x>2 \\ ax^{2}+b, если\ x<2 \end{cases} \quad x=\begin{cases} ab+2, \ если\ a\geq b \\ a/b+2, если\ a<b \end{cases}$	a=4.1 b=3.7	*x, y.*
13	$z=\begin{cases} \sin x+a, \ если\ x<a \\ \cos \pi x-b, \ если\ a\leq x\leq b \\ tg(x^{2}), \ если\ x>b \end{cases}$ $x=\begin{cases} (a-1)(b-2), \ если\ a>b-1 \\ (a+1)/(b+2), \ если\ a\leq b-1 \end{cases}$	a=2.7 b=3.5	*x, z.*
14	$y=\begin{cases} 1+x+x^{2}, если\ x\geq 5 \\ 1/x, \ если\ -5<x<5 \\ 1/x^{2}, \ если\ x\leq -5 \end{cases}$ $x=\begin{cases} c+0.7, \ если\ c\leq d \\ d-0.5, если\ c>d \end{cases}$	c=2.5 d=1.9	*x, y.*
15	$z=\begin{cases} x+y, если\ xy<a \\ xy, если\ a\leq xy\leq b \\ x-y, \ если\ xy>b \end{cases}$ $y=\begin{cases} ax/b, \ если\ x<3 \\ ab/x, \ если\ x\geq 3 \end{cases}$	a=1.5 b=1.9 x=2.3	*z, y.*
16	$z=\begin{cases} x^{2}+y^{2}, если\ y>x+1 \\ x^{2}\ln y, если\ y=x+1 \\ x^{2}-y^{2}, если\ y<x+1 \end{cases}$ $y=\begin{cases} x+a, \ если\ x=a \\ x/a, \ если\ x\neq a \end{cases}$	a=3.4 x=1.4	*z, y.*
17	$z=\begin{cases} ax+bx, \ если\ x<1 \\ ax/by, если\ 1\leq x\leq 9 \\ ax-by, \ если\ x>9 \end{cases}$ $y=\begin{cases} \sqrt{ab}, \ если\ a\leq b \\ \sqrt{a+b}, если\ a>b \end{cases}$	a=3.2 b=2.4 x=4.1	*z, y.*

Продолжение таблицы к заданию 1

1	2	3	4
18	$z=\begin{cases} ax+by, & \text{если } a\le x\le b \\ \ln(bx)+ay, & \text{если } -b\le x\le -a \\ xy+1, & \text{в остальных случаях} \end{cases}$ $y=\begin{cases} a^2+b^2, & \text{если } a\ne b \\ ab-1, & \text{если } a=b \end{cases}$	a=2.7 b=4.3 x=3.1	*z, y.*
19	$y=\begin{cases} a+\sqrt{cx}, & \text{если } x<3 \\ b+\sin \pi x, & \text{если } 3\le x\le 5 \\ c-\cos ax, & \text{если } x>5 \end{cases}$ $x=\begin{cases} c+ab, & \text{если } b\le a-1 \\ c-ab, & \text{если } b>a-1 \end{cases}$	a=3.7 b=2.9 c=0.3 d=4.5	*x, y.*
20	$z=\begin{cases} ax^2+bx+c, & \text{если } x<2 \\ bx^2+ax+c, & \text{если } 2\le x\le 4 \\ cx^2+ax+b, & \text{если } x>4 \end{cases}$ $x=\begin{cases} (a+b)+c, & \text{если } a+1<cb \\ (a-b)/(a-c), & \text{если } a+1\ge cb \end{cases}$	a=4.3 b=5.1 c=1.4	*x, z.*

Задание №2. Организация циклов с известным числом повторений

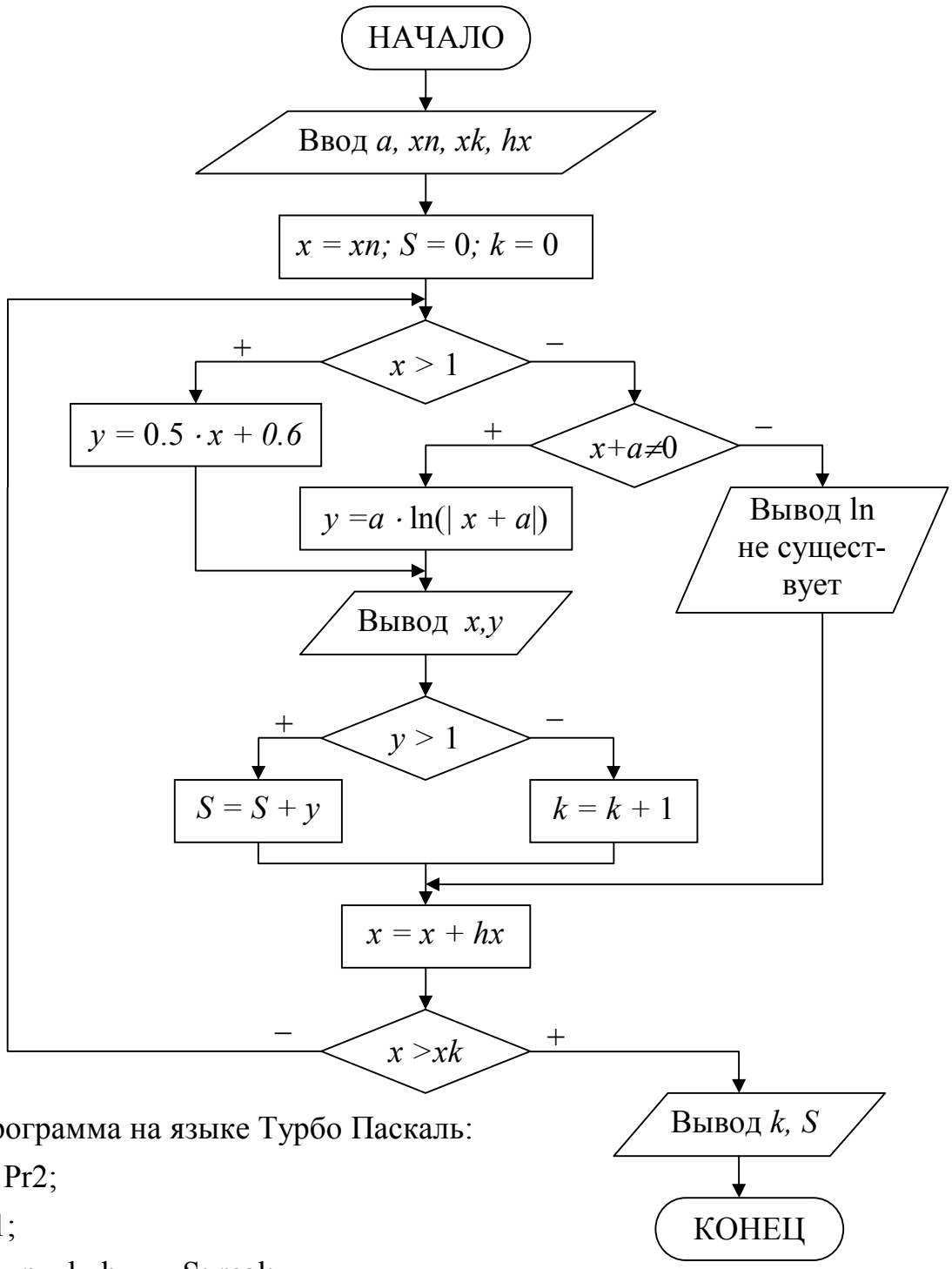
Задание. Составить блок-схему алгоритма и программу решения поставленной задачи, которая в соответствии с исходными данными вычисляет значения заданных выражений.

Пример. Составить алгоритм и программу для вычисления значений ***y*** при всех возможных значениях ***x***, которые лежат в интервале от ***xn*** до ***xk*** с шагом ***hx***.

$$y=\begin{cases} 0.5x+0.6, & \text{если } x>1 \\ a\cdot\ln(|x+a|), & \text{если } x\le 1 \end{cases}$$

Вычислить сумму значений ***y*** > 1 и кол-во ***y*** ≤ 1.

Блок-схема алгоритма:



Программа на языке Турбо Паскаль:

```
Program Pr2;
Label m1;
Var a, x, xn, xk, hx, y, S: real;
    k: integer;
Begin
 Write('Ввод a, xn, xk, hx'); Readln(a,xn,xk,hx);
 x:=xn; S:=0; k:=0;
 Repeat
  If x > 1 Then   y := 0.5*x + 0.6
  Else  If x + a < > 0 Then  y := a * ln(abs(x + a))
        Else  Begin  Writeln('ln не существует');  Goto m1; End;
```

```
  writeln('x=', x : 6 : 2, ' y=', y : 6 : 2);
  If y > 1 Then S := S + y Else k := k + 1;
m1:
 x := x + hx;
Until x > xk;
writeln('k=', k, ' S=', S : 6 : 2);
End.
```

Варианты заданий.

№ п/п	Модель	Исходные данные	Выводимые данные
1	**2**	**3**	**4**
1	$Z=\begin{cases}2x^3+3, если & x\ge 5\\ 7x+6, \ если & 1\le x<5\\ -2/x^3, \ если & x<1\end{cases}$	$0\le x\le 10$ hx=1	*Z, x.* *Количество* $Z\in$ [-1;1]. *Сумма* $Z\notin$ [-1;1].
2	$y=\sum_{i=1}^{N}2/\sin ix \quad N=\begin{cases}10, если\ a<0\\ 5, \ если\ a=0\\ 20, если\ a>0\end{cases}$	x=0.8 a	*y, N, a, x*
3	$y=\begin{cases}2\sin^2 x+x^2, \ если & x>0\\ x^2-1/x, \quad если & x\le 0\end{cases}$	-2 ≤ x ≤ 2 hx=0.2	*x, y. Среднеарифметическое положительных значений y.*
4	$F=\begin{cases}-4, \quad если & x<-1\\ x^2+3/x+4, если & -1<x<1\\ (x+4)^2, \quad если & x>1\end{cases}$	3 ≤ x ≤ 5 hx=0.1	*F, x. Сумма, количество положительных значений F.*
5	$F=N! \quad N=\begin{cases}5, \ если\ x>0\\ 7, \ если\ x=0\\ 10, если\ x<0\end{cases}$	-2≤ x ≤ 2 hx=1	*x, N, F*
6	$F=\frac{(N-k)b}{(N+k)a}$, $y=\begin{cases}ax+b, \quad если\ x<0.5\\ \ln x+e^x, если\ x=0.5\\ x+a/b, \quad если\ x>0.5\end{cases}$	*a,b* 0 ≤ x ≤ 2 hx=0.2	*y, x, N, k.* *F,* *где N – к-во y>0,* *k – к-во y≤0.*

Продолжение таблицы к заданию 2

1	2	3	4
7	$y=\frac{3\sin(\omega\pi+x)}{2+\cos(x-\omega\pi)}$ $\omega=\begin{cases}\pi-\cos x, если\ \|x\|\le\pi/4\\ \pi+\cos x, если\ \|x\|>\pi/4\end{cases}$	$-\frac{\pi}{3}\le x\le\frac{\pi}{3}$ $hx=\frac{\pi}{20}$	*x, ω, y.* *Произведение (P).* $y\in[0;5]$
8	$S=\sum_{i=1}^{20}\ln\left\|\frac{y}{i}\right\|$, *где* $y=\begin{cases}2-1/x, если \cos x\le 0.3\\ x\cdot tgx, \ если \cos x\ge 0.3\end{cases}$	*x*	*x, y, S*
9	$y=t^{-x}+5$ $t=\begin{cases}0.7-1/x, если\ x\ge 0\\ x+0.3, \quad если\ x<0\end{cases}$	$-5\le x\le 5$ $hx=1$	*x, t, y.* *Количество y>t.*
10	$x=\begin{cases}\sum_{i=1}^{10}t^{-i}, \ если \ln t<1.5\\ t^2-\sqrt{t}, если \ln t\ge 1.5\end{cases}$ $z=\begin{cases}ln\,x+1, если\ x>0.5\\ x, если\ x\le 0.5\end{cases}$	*t*	*x, z, t*
11	$Q=\begin{cases}\sum_{i=1}^{10}(1+\cos 0.1i), если \quad e^{-a}<0.1\\ \pi/\sin 0.5a, \quad если \quad e^{-a}\ge 0.1\end{cases}$	*a*	*a, Q*
12	$R=\sqrt{\sum_{y>z}(y-z)^2}$ $y=sin^2x+0.5cosx^2$ $Z=\begin{cases}\cos^2\frac{\pi}{4}x, если \quad \|y\|>x^2\\ 1+1/x, \quad если \quad \|y\|\le x^2\end{cases}$	$0\le x\le 2$ $hx=0.1$	*Z, y, x, R.* *Количество слагаемых в R.*
13	$S=\sum_{i=1}^{5}Z, \quad где \quad y=i^2-i-10$ $Z=\begin{cases}y+\frac{\pi}{2}\sin\frac{\pi}{2}y, если \quad y<0\\ 1, \qquad если \quad y\ge 0\end{cases}$	$1\le i\le 10$ $hx=1$	*i, y, Z, S.* *Количество y>0*

Продолжение таблицы к заданию 2

1	2	3	4
14	$S=\sum p; A=\prod q$ $q=0.5\sin \pi x$ $p=\begin{cases}0.5x\cos x, если & q>0.5\\ 2x\sin x, если & q\le 0.5\end{cases}$	$0\le x\le 2$ hx=0.2	*x, q, P, S, A.* *Количество p*<0.
15	$S=\sum_{y>0} y; P=\prod_{y<0} y$ $y=\begin{cases}e^{\sin x}, & если\ a^2x<b^3\\ (x^2-a)/\sin x, & если\ a^2x=b^3\\ tg4.5x, & если\ a^2x>b^3\end{cases}$	*a,b* $\frac{\pi}{2}\le x\le 2\pi$ $hx=0.1\pi$	*x, y, S, P.*
16	$z=2.5e^{xy}-1.8$ $y=\begin{cases}(x-1.7)^2, & если\ a\le x\le b\\ 1-\sqrt[3]{x}, & если\ c\le x\le d\\ -1.2x, & в\ остальных\ случаях\end{cases}$	*a,b,c,d* $0.1\le x\le 1$ hx=0.1	*x, y, z.* *Сумма вычисленных z.*
17	$F=\begin{cases}Z, если\ Z>0\\ 0, если\ -1\le Z\le 0\\ Z^2, если\ Z<-1\end{cases}$ $Z=x^3+5x,\quad y=F+0{,}38Z$	$-1\le x\le 5$ hx=0.2	*y, F, x, Z.* *Количество y>Z.*
18	*Определить действительные корни уравнения* $ax^2+bx+c=0$	*a,c* $-4\le b\le 5$ hb=1	*b и соответствующие действительные корни уравнения.*
19	$f=y^2+x$ $y=\begin{cases}1.7+b/\sin^2 x, если\ x<3\\ 8.5x-b, & если\ x\ge 3\end{cases}$	*b* $0\le x\le 5$ hx=0.5	*x, y, f* *количество f*>0 *и f*<0
20	$y=\begin{cases}tgZ, & если\ Z\ge 1.4\\ Z^2/i, & если\ Z<1.4\end{cases}$ *Z=Ln(i)*	$1\le i\le 10$ hi = 1	*i, y, Z. Произведение и количество положительных y.*

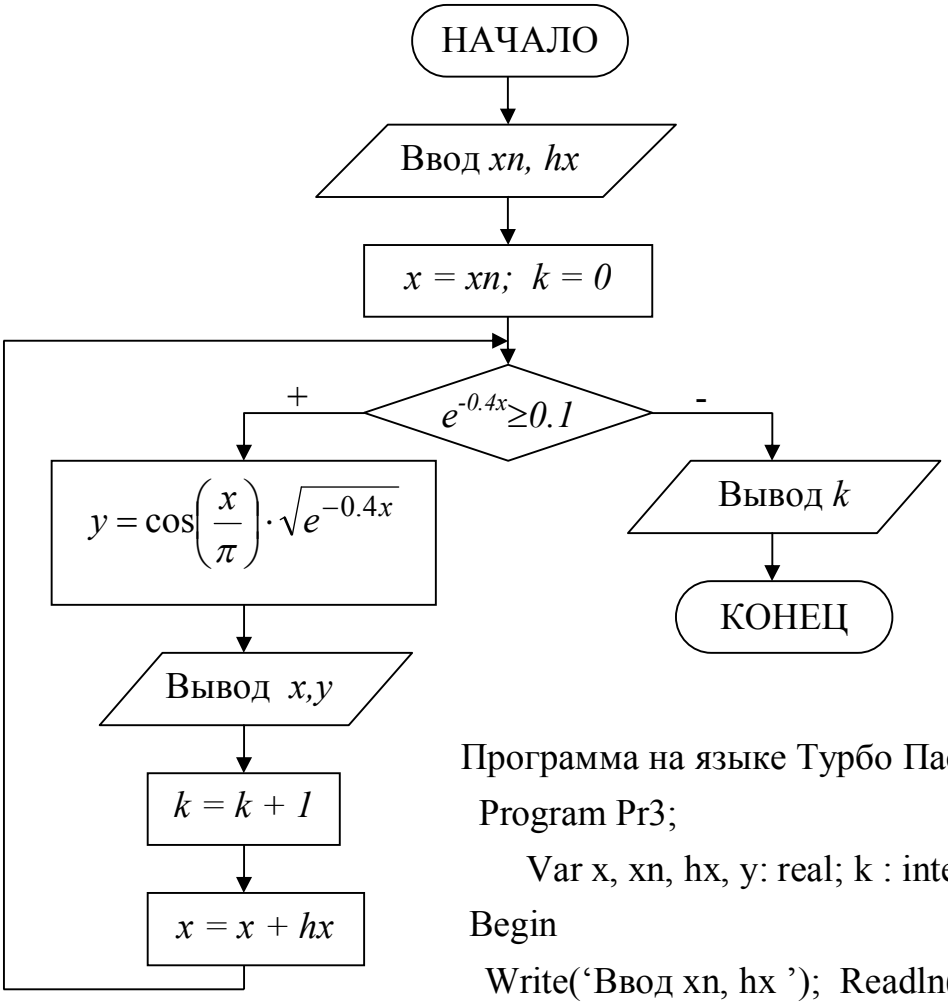
Задание № 3. Организация циклов с неизвестным числом повторений

Задание. Составить блок-схему алгоритма и программу решения поставленной задачи, которая в соответствии с исходными данными вычисляет значения заданных выражений.

Пример. Составить алгоритм и программу для вычисления ***y,*** при всех значениях ***x***, начинающихся с начального ***xn*** > 0, и изменяющихся с шагом 0.6. Вычислять ***y***, пока подкоренное выражение ≥0.1. Найти количество вычисленных ***y***.

Блок-схема алгоритма:

$$y = \cos\left(\frac{x}{\pi}\right) \cdot \sqrt{e^{-0.4x}}$$



Программа на языке Турбо Паскаль:

```
Program Pr3;
    Var x, xn, hx, y: real; k : integer;
Begin
 Write('Ввод xn, hx ');  Readln(xn, hx);
  x:=xn; k := 0;
 While  Exp(-0.4 * x) >= 0.1 do Begin
     y := cos(x / Pi ) * Sqrt(Exp(-0.4 * x));
     writeln('x=', x : 6 : 2, ' y=', y : 6 : 2);
     k := k + 1;     x := x + hx;   End;
 writeln('k=', k);  End.
```

Варианты заданий.

№ п/п	Модель	Исходные данные	Выводимые данные
1	**2**	**3**	**4**
1	$F=\sqrt{1+0.2\frac{q}{2q^2-1}-\frac{1}{q+1}}$ *Считать F до тех пор, пока подкоренное выражение* >0.	$q \le 3$ *hq*=-0.2	*F, q.* *Количество вычисленных F.*
2	$y=7.35a+\sqrt{0.2\frac{a^3}{2a^2-1}};S=\sum y$ *Считать y до тех пор, пока выражение под знаком корня* > 1.	$a \le 7$ *ha*=-0.5	*y, a, S.* *Количество вычисленных y.*
3	$F=b^2\sqrt{0.1+x^2}+\frac{3}{b\sqrt{0.1+x}}$ $S=\sum F$ *Считать F пока не превысит A.*	*b, A* *x*≥0 *hx*=0.5	*x, F, S.* *Количество слагаемых в сумме.*
4	$F=2.72y+2Z^2\sin(x+y)$ $x=a^2-\sqrt{a}$ $y=\begin{cases}\frac{x^{n+1}}{n+1}, \text{если } n\neq -1\\ \ln x, \text{ если } n=-1\end{cases}$ *Считать F до тех пор, пока F остаётся меньше* 100.	*a, n,* $Z \ge 0.4$ *hz* = 0.5	*x, y, F, Z .* *Количество вычисленных F.*
5	$f=t^3\ln Z+1$ $Z=\begin{cases}\frac{1.5t^2}{2a}+c, t\le 3\\ 6.5t+c, t>3\end{cases}$ $t=\sin^2 a+\sqrt[3]{a}, P=0.345+\ln(f+b^3)$ *Считать P, пока* $(f+b^3)>0$.	*c,a* $b \le 2$ *hb*=-0.2	*t, Z, f, b, P.* *Количество вычисленных P.*
6	$y=\sin^2 x+\ln(x+\sin x)$ $P=Пy$ *Считать P, до тех пор, пока станет* > *Q.*	*Q,* $x \ge 1$ *hx*=0.5	*x, y, P.* *Сумма и количество положительных значений y.*
7	$x=2ab\lvert\sin \pi t\rvert$ $Z=\sqrt{x+t}$ *M=K!, где K- количество Z.* *Считать Z, пока выражение* $x+t\ge 0$.	*a, b,* $t \le 5$ *ht*=-0.5	*t, x, Z, M, K.*

Продолжение таблицы к заданию 3

1	2	3	4
8	$y = a\cos a + \ln\left\|\sin\frac{x}{3}\right\|$ $F = \sum_{-3\le y\le 3} y$ *Считать F, пока значение F<Q.*	*x, Q,* *a*≥0 *ha*=0.5	*y,a,F.* *Количество слагаемых в сумме.*
9	$F=5.37x + \ln(x^3+x^2+x)$ $P=\prod F$ *Считать F до тех пор, пока выражение под знаком логарифма* > 0.	$x \le 3$ *hx*=-0.1	*F, x, P.* *Количество сомножителей в P.*
10	$Q = \frac{a+b}{2a-b}(a+c)\sin(x+a)$ $F = \prod Q$ *Считать F до тех пор, пока* $F\in[-2;5]$.	*b, c, x,* *a*≥0 *ha*=0.5	*a, Q, F.* *Количество сомножителей в F.*
11	$Z = \frac{x^4}{36.04x^3 - \frac{0.98\sin^3 x}{15.1x - \ln x}}$ *Считать Z до тех пор, пока оно остается* < *Q.*	*Q,* *x*≥1, *hx*=0.1	*x, Z.* *Количество вычисленных Z, и сумма первых пяти Z.*
12	$Z = 2.33q + \sqrt{0.2\frac{q^3}{2q^2-1}}$ *Считать Z до тех пор, пока подкоренное выражение положительно.*	$q \le 10$ *hq*=-0.5	*q, z.* *Количество и сумма вычисленных значений Z.*
13	$y = \sqrt[5]{\pi a^2} + \frac{1}{b} + \frac{b-\frac{a}{2}}{a+b}$ *Считать y до тех пор, пока подкоренное выражение* $\le C$	*b, C.* *a*≥0 *ha*=0.1	*a, y.* *Вычислить K=N!, где N - кол-во вычисленных y.*
14	$Z = \frac{x}{\sqrt{a^2+b^2}} + tgx^3 \qquad y = \prod Z.$ *Считать y до тех пор, пока оно остается меньше* 100.	*a, b,* *x*≥5 *hx*=0.3	*Z, x, y.* *Количество Z > b.*

Продолжение таблицы к заданию 3

1	2	3	4
15	$A = \sin^2 b + \cos(b - \pi) + 1$ $F = \prod A$ *Считать F до тех пор, пока F остается* < 10.	$b \geq 0$ $hb = 0.1$	*b, A, F.* *Количество A > 0.*
16	$y = 8.36 \sin x + \sqrt{a^2 + \frac{\pi}{2}}$ *Считать y до тех пор, пока* $a^2 + \frac{\pi}{2}$ *превысит Q.*	*x, Q,* $a \geq 0$ $ha = 0.4$	*a, y.* *Количество (N) вычисленных y.* *K=N!.*
17	$C = \frac{1 - \sin b}{\ln(b^5 - b^2 + b)}$ $F = \sum C$ *Считать C до тех пор, пока выражение под знаком логарифма* >1.	$b \leq 3$ $hb = -0.2$	*b, C, F.* *Количество вычисленных C.*
18	$F = \sqrt{1 + \sin(q/2) - \frac{1}{q+1}}$ $y = \sum_{F>1} F$ *Считать F до тех пор, пока подкоренное выражение* >0.	$q \leq 2$ $hq = -0.1$	*F, q.* *Количество слагаемых в сумме.*
19	$F = 2.72y + 2Z^2 \sin(x + y)$ $x = a^2 - \sqrt{a}$ $y = \begin{cases} \frac{x^{n+1} - 1}{n+1}, если\_n \neq -1 \\ \ln x, если\_n = -1 \end{cases}$ *Считать F до тех пор, пока F остаётся меньше* 100.	*a, n* $Z \geq 0.4$ $hZ = 0.5$	*Z, F, x, y.* *Количество вычисленных F.*
20	$y = \ln 2x - x^2$ $Z = \frac{\sum y}{N}$ *Считать y до тех пор, пока выражение под знаком логарифма* >0.	$x \leq 10$ $hx = -0.5$	*x, y, Z.* *N – количество слагаемых в сумме.*

Задание № 4. Организация вложенных циклов

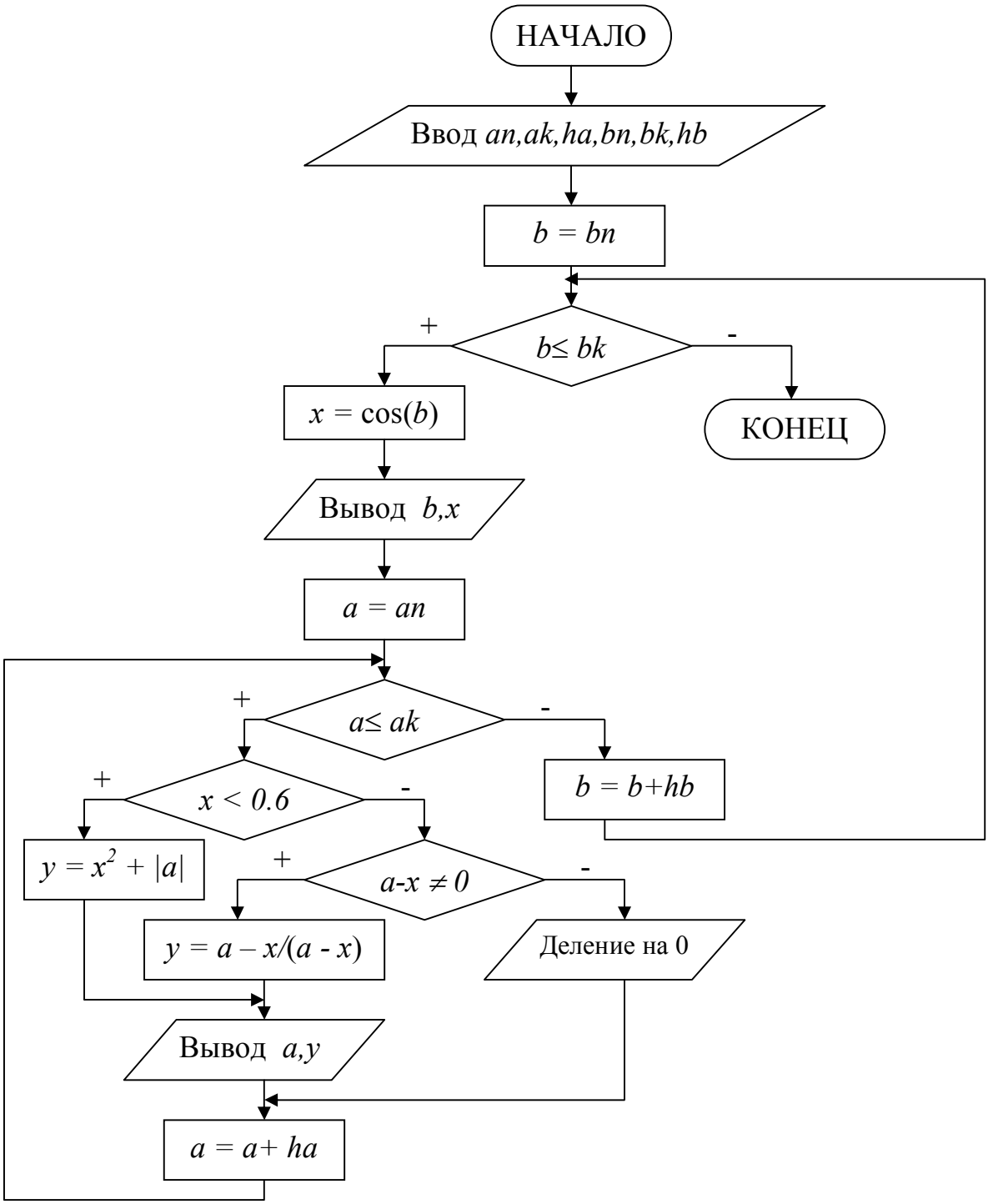
Задание. Составить блок-схему алгоритма и программу решения поставленной задачи, которая в соответствии с исходными данными вычисляет значения

заданных выражений.

Пример. Составить алгоритм и программу для вычисления значений ***x*** и ***y*** при всех возможных значениях ***a*** и ***b***, которые лежат в интервале от ***an*** до ***ak*** с шагом ***ha*** и от ***bn*** до ***bk*** с шагом ***hb***, соответственно.

$$y=\begin{cases} x^2+|a|, & \text{если} \quad x<0.6 \\ a-x/(a-x), & \text{если} \quad x\geq 0.6 \end{cases}$$

Блок-схема алгоритма:



Программа на языке Турбо Паскаль:

```
Program Pr4;
Label m1;
```

```
Var a, an, ak, ha, b, bn, bk, hb, x, y: real;
Begin
 Write('Ввод an, ak, ha, bn, bk, hb'); Readln(an, ak, ha, bn, bk, hb);
 b:=bn;
 While b <= bk do Begin
   x := cos(b);   writeln('b=', b : 6 : 2, ' x=', x : 6 : 2);
   a := an;
  While a <= ak do Begin
    If  x < 0.6 Then  y := Sqr(x) + Abs(a)
    Else If a – x < > 0 Then y := a – x / (a – x)
        Else  Begin  Writeln('Деление на 0');  Goto m1; End;
    writeln('a=', a : 6 : 2, ' y=', y : 6 : 2);
m1:  a := a + ha;   End;
 b := b + hb; End;
End.
```

Варианты заданий.

№ п/п	Модель	Исходные данные	Выводимые данные
1	2	3	4
1	$y=\begin{cases} e^{\sin x}, & если\ a^2x<b^3 \\ (x^2-a)\sin x, & если\ a^2x=b^3 \\ tg^2 4.5x, & если\ a^2x>b^3 \end{cases}$	x, $1 \le a \le 2$; ha=0.1 $-3 \le b \le 1$, hb=1	a,b,y
2	$z=\begin{cases} ax-\dfrac{\sqrt{ax}}{0{,}2x+0{,}5a}, & если\ x\ge 0 \\ 2a+x^2+0{,}7, & если\ x<0 \end{cases}$ $x=\dfrac{b}{a+0{,}1}$	$0.6 \le a \le 1.2$ $-0.3 \le b \le 1.2$ ha=0.2; hb=0.3	a, b x, z,
3	$z=\begin{cases} y^2+\dfrac{a+y^2}{ay}, & если\ y\ge 1 \\ y/a+\sqrt{a+3y}, & если\ y<1 \end{cases}$ $y=\dfrac{x+a}{2x}$	$3 \le a \le 6$ ha = 1 $0.2 \le x \le 1$ hx = 0.2	a, x, y, z
4	$z=\begin{cases} \dfrac{x^2}{x+a}+\sqrt{x}, & если\ a\ge 1 \\ \sqrt{ax}+3x, & если\ a<1 \end{cases}$ $a=\sqrt{2x^2+0{,}2c}$	$1.2 \le c \le 2$ $2 \le x \le 8$ hc = 0.2; hx = 2	c, x, a, z

Продолжение таблицы к заданию 4

1	2	3	4
5	$z=\begin{cases}\dfrac{px-5}{bx+0{,}5}-\sqrt{bx+5}, \text{если } bx\ge 5\\ \dfrac{2p^{2}}{x^{2}+3}+\sqrt{5+2bx}, \text{если } bx<5\end{cases}$ $x=3k+2$	$1.4\le b\le 2.6$ $hb=0.3$; $p=0.4$ $1\le k\le 7$; $hk=3$	b, k, x, z
6	$z=\begin{cases}\dfrac{ax^{2}+b}{bx+a}+\sqrt{ab+x}, \text{если } x\ge a\\ \dfrac{bx-a}{x^{2}}-\sqrt{x+a}, \text{если } x<a\end{cases}$ $x=k^{2}-2$	$3\le k\le 15$ $hk=4$ $2\le b\le 4$ $hb=0.5$, a	k, b, x, z
7	$z=\begin{cases}\sqrt{a^{2}+x^{2}}+\sqrt{x/(a+0.2x)}, \text{если } x\ge 0\\ \sqrt{a^{2}+x^{2}}-\sqrt{x/(a+2x)}, \text{если } x<0\end{cases}$ $x=k^{2}+k+0.1$	$2.2\le a\le 4.2$ $ha=0.5$ $-0.5\le k\le 0.4$ $hk=0.4$	a, k, x, z
8	$z=\begin{cases}ax^{2}+ax-\sqrt{\dfrac{b}{x+0.2}}, \text{если } x\ge 0.2\\ \dfrac{ax^{2}-ax}{a+x}\sqrt{\dfrac{b}{x+0.2}}, \text{если } x<0.2\end{cases}$ $x=(2t^{2}+0.3)/2$	$b=7$ $0.5\le a\le 2$ $-1.2\le t\le 0.4$ $ha=0.5$; $ht=0.4$	a, t, x, z
9	$z=\begin{cases}ax+1-\dfrac{a^{2}}{x}, \text{если } x<6\\ \dfrac{x-a}{\sqrt{ax}}+2a, \text{если } x\ge 6\end{cases}$ $x=0.5t^{2}-2$	$1\le a\le 2$ $ha=0.5$ $-5\le t\le 7$ $ht=3$	a, t, x, z
10	$z=\begin{cases}kx+\sqrt{2x+b}, \text{если } b\ge 0.5\\ \dfrac{\sqrt{kb}}{bk+3}-k^{2}x, \text{если } b<0.5\end{cases}$ $b=\dfrac{k+0.7}{3}$	$4\le x\le 6$; $hx=1$ $1\le k\le 6.1$ $hk=1.7$	x, k, b, z
11	$z=\begin{cases}\dfrac{a}{x^{2}+1.5}+\sqrt{a+x}, \text{если } x\ge 2\\ 2\sqrt{a}-\dfrac{x}{a}, \quad \text{если } x<2\end{cases}$	$-3\le x\le 3$ $hx=2$ $1\le a\le 2$ $ha=0.5$	a, x, z

Продолжение таблицы к заданию 4

1	2	3	4
12	$z=\begin{cases} x^2\left(\sqrt{c+2}-ic\right), \text{если } x>0 \\ \dfrac{b}{\sqrt{i^2+1.7}}, \quad \text{если } x\le 0 \end{cases} \quad x=i^2-0.7$	$3 \le c \le 5$ $hc = 0.5$ $-0.5 \le i \le 0.5$ $hi = 0.5$	*c, i, x, z*
13	$z=\begin{cases} a^2-x^2-\dfrac{x}{a+1}, \text{если } x>2 \\ \dfrac{7x-a}{x^2}+0.6a^2, \text{если } x\le 2 \end{cases}$ $x=\sqrt{k^2+0.6}$	$-5 \le a \le -3$ $ha = 1$ $-1 \le k \le 2$ $hk = 1.5$	*a, k, x, z*
14	$z=\begin{cases} x^2-\dfrac{b}{\sqrt{b^2-x}}, \text{если } x<0.5 \\ \sqrt{x}\left(b+3x^2\right), \text{если } x\ge 0.5 \end{cases} \quad x=\dfrac{t^2}{2+t}$	$4 \le b \le 8$ $hb = 1$ $1 \le t \le 2.5$ $ht = 0.5$	*b, t, x, z*
15	$z=\begin{cases} \dfrac{x-a}{\sqrt{x^2+1}}, \quad \text{если } x>5 \\ \dfrac{x-3}{a}+\sqrt{a^2+x^2}, \text{если } x\le 5 \end{cases}$ $x=3+0.5t$	$5 \le a \le 7$ $ha = 1$ $0.5 \le t \le 2$ $ht = 0.5$	*a, t, x, z*
16	$z=\begin{cases} \sqrt{x^2+\dfrac{a^2}{4}}+x^3, \text{если } x<1.6 \\ a+\dfrac{x}{\sqrt{a}-x}, \text{если } x\ge 1.6 \end{cases}$ $x=\sqrt{4+t}$	$1.5 \le a \le 2.5$ $ha = 0.5$ $1.5 \le t \le 4.5$ $ht = 1.5$	*a, t, x, z*
17	$z=\begin{cases} \sqrt{x+a}\left(x^2-1\right), \text{если } x>0 \\ \dfrac{\sqrt{a}+1-x}{x^2+a}, \text{если } x\le 0 \end{cases} \quad x=\dfrac{k+1.5}{k}$	$2 \le a \le 3$ $1.2 \le k \le 2.8$ ha=0.5 hk=0.4	*a, k, x, z*
18	$z=\begin{cases} bx+\dfrac{25}{\sqrt{b+x^2}}, \text{если } x<3 \\ b\sqrt{\dfrac{b}{x}+3}, \text{если } x\ge 3 \end{cases}$ $x=0.3t^2$	$4 \le t \le 8.5$ $3.7 \le b \le 4.7$ ht=2.5 hb=0.5	*t, b, x, z*

Продолжение таблицы к заданию 4

1	2	3	4
19	$z=\begin{cases}\dfrac{a}{\sqrt{x^2+a^2}}-bx, \text{если } x>a\\ \dfrac{a}{\sqrt{x^2+a^2}}+\dfrac{b}{x}, \text{если } x\le a\end{cases}$	a=10.3 $0.3\le b\le 1.3$ hb=0.5 $1\le x\le 3$; hx=0.5	b, x, z
20	$z=\begin{cases}\dfrac{ax^2+b}{cx+\sqrt{x+100}}, \text{если } x\ge 0\\ \dfrac{ax^2-b}{1+cx+\sqrt{x+100}}, \text{если } x<0\end{cases}$ $x=(i-a)/i$	a, c $1\le b\le 3$; $hb=1$ $1\le i\le 3$; hi=1.5	b, i, x, z

Задание № 5. Обработка одномерных массивов

Задание. Составить блок-схему алгоритма и программу решения поставленной задачи, которая в соответствии с исходными данными вычисляет значения заданных выражений.

Пример. Составить алгоритм и программу формирования массива Y на основе исходного массива X размерностью ***N***. Определить максимальный положительный элемент массива Y. Элементы массива Y вычисляются по формуле:

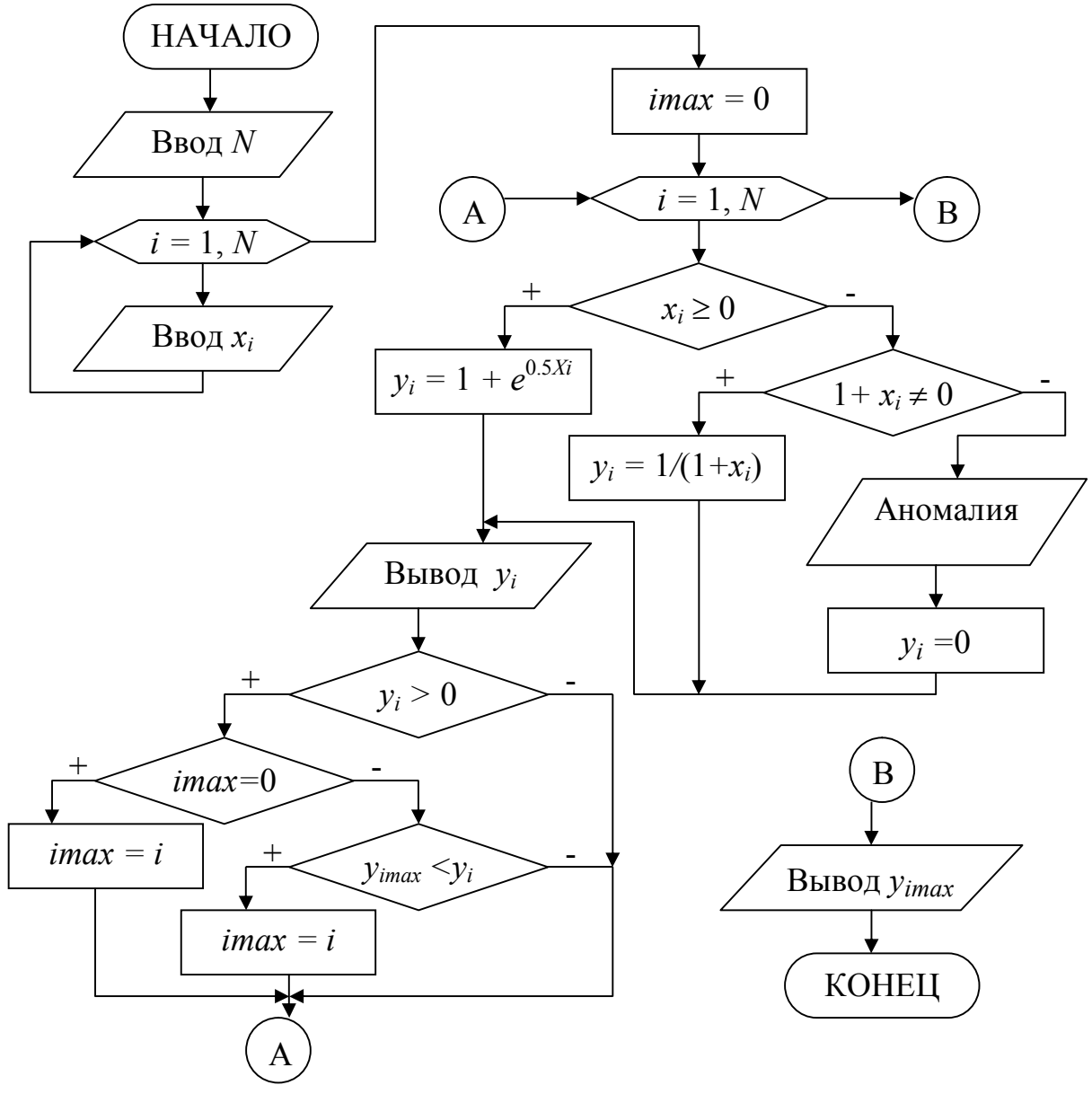
$$y_i=\begin{cases}1+e^{0.5X_i}, & \text{если } x_i\ge 0\\ 1/(1+x_i), & \text{если } x_i<0\end{cases}$$

Программа на языке Турбо Паскаль:

```
Program Pr5;
Var x, y: array [1..20] of real;
    i, N, imax: byte;
Begin
 Write('Ввод N'); Readln(N);
 for i:=1 to N do begin
  Write('Ввод x[', i, '] ='); Readln(x[i]);
 end;
 imax := 0;
 for i:=1 to N do begin
  if x[i] >= 0 then y[i] := 1+exp(0.5*x[i])
```

```
  else
     if 1+x[i] <> 0 then y[i] := 1/(1+x[i])
     else  begin   writeln('Аномалия');  y[i] := 0
           end;
    writeln('y[', i, ']=', y[i] : 6 : 2);
    if y[i] > 0 then
      if imax = 0 then  imax := i
      else  if y[imax] < y[i]   then  imax := i;
  end;
  writeln('y[', imax, ']=', y[imax] : 6 : 2);
End.
```

Блок-схема алгоритма:



Варианты заданий.

№ п/п	Модель	Исходные данные	Выводимые данные
1	**2**	**3**	**4**
1	$y_i = \begin{cases} 25x_i + 2, & \text{если } 2 < x_i \le 25 \\ 5\cos^2 x_i, & \text{если } x_i > 25 \\ 1/x_i^3, & \text{в остальных случаях} \end{cases}$	*Массив* X $i = 1 \div 10$	*Массив* Y *Значение наибольшего отрицательного элемента массива* Y
2	$z_i = \begin{cases} y_i - 0.3\,y_i^2 / y_i + 1, & \text{если } y_i > 1 \\ 0.5\cos \pi y_i, & \text{если } \lvert y_i \rvert \le 1 \\ 2\sin(\cos \pi/2\, y_i), & \text{если } y_i < -1 \end{cases}$	*Массив* Y $i = 1 \div 15$	*Массив* Z. *Максимальный элемент* Z_{max} *и номера элементов массива* Z *меньших* $0.5Z_{max}$
3	$z_i = 2\sin^2 \frac{\pi}{3} x_i + 3{,}5x_i^3$ $y_i = \begin{cases} z_i + \lvert z_i \rvert, & \text{если } z_i < -1 \\ 1 + e^{-z_i}, & \text{если } z_i > 3 \\ \cos z_i + z_i^2, & \text{если } -1 \le z_i \le 3 \end{cases}$	*Массив* X $1 \le x_i \le 10$ hx_i=1	*Массивы* Z, Y. *Сумма и количество положительных элементов массива* Y. *Среднее арифметическое отрицательных элементов массива* Y.
4	$y_i = \sin^2 x_i + \sqrt{1 + \cos^2 x_i^2}$ $z_i = \begin{cases} \ln(\cos^2 \frac{\pi}{4} x_i + 0.01), & \text{если } \lvert y_i \rvert > x_i^2 \\ 1 + x_i - x_i^2, & \text{если } \lvert y_i \rvert \le x_i^2 \end{cases}$	*Массив* X $-1 \le x_i \le 11$ hx_i=2	*Массивы* Z, Y. *Порядковый номер и значение первого положительного числа в массиве* Z.
5	$y_i = \begin{cases} \frac{\sqrt{x_i}\sin \pi x_i}{x_i + e^{x_i}}, & \text{если } x_i > 1.5 \\ 2x_i + \sqrt{e^{x_i}}, & \text{если } x_i \le 1.5 \end{cases}$ *Все отрицательные элементы массива* Y *заменить нулями, а нулевые элементы заменить значением элемента* x_i	*Массив* X $i = 1 \div 15$	*Массив* Y *до и после замены.* *Среднее арифметическое массива* Y *до и после замены.*

Продолжение таблицы к заданию 5

1	2	3	4
6	$y_i = \begin{cases} \sin x_i^2 + \sqrt{\lvert x_i \rvert + 1}, \text{если} \lvert x_i \rvert < 2 \\ e^{0.5x_i} + \ln(x_i^2 + 1), \text{если} \lvert x_i \rvert \geq 2 \end{cases}$ $S = \frac{1}{15}\sum_{i=1}^{15} y_i$	*Массив* X i = 1÷15	*Массив* Y. *Значение и номер эл-та* y_i, *наиболее отличающегося от* ***S***.
7	$a_i = 2\sin x_i + 0{,}3$ $b_i = \begin{cases} \sqrt{a_i}, \text{ если } x_i < 1 \\ 2.5a_i - \sqrt[3]{a_i}, \text{ если } x_i \geq 1 \end{cases}$ $c_i = max(a_i, b_i) - min(a_i, b_i)$	*Массив* X i = 1÷10	*Массивы* A, B, C. *Максимальный элемент массива* C *среди четных элементов.*
8	$m_i = \begin{cases} 1 + arctg \frac{x_i}{1 + \sqrt{x_i}}, \text{если } x_i > 0.147 \\ \sin x_i^{2x_i}, \text{ если } x_i \leq 0.147 \end{cases}$ $S = \frac{1}{N}\sum_{i=1}^{N} M_i \qquad P = \sqrt[N]{\prod_{i=1}^{N} M_i}$	*Массив* X i = 1÷25	*Массив* M. *Разность между S и P.*
9	$p_i = \begin{cases} \lvert 1 + x_i \sin x_i \rvert, \text{если } x_i \geq 0.2 \\ \sqrt{1 + 2x_i^3}, \quad \text{если } x_i < 0.2 \end{cases}$ *Если* $maxP_i$ *меньше суммы всех остальных элементов, то присвоить этому элементу значение* 0.	*Массив* X -0.4≤ x_i ≤1.2 hx_i=0.2	*Массив* P, $maxP_i$.
10	$p_i = \begin{cases} y_i + \pi \cos \pi y_i, \text{ если } x_i > 1 \\ 1 + \sqrt{\lvert y_i \rvert + 1}, \text{ если } x_i \leq 1 \end{cases}$ $x_i = y_i^2 + 2y_i + 3$	*Массив* Y i = 1÷11	*Массивы* X, P. *Значения и номера мин. и макс. по модулю элементов* P.
11	$y_i = \frac{3\sin(\varpi t + x)}{2 + \cos(x - \varpi t)}$ $\varpi = \begin{cases} \frac{\pi}{2} - 2x, \quad \text{если } x \leq 2 \\ \pi - 2x, \text{ в остальных случаях} \end{cases}$	t $-\frac{\pi}{3} \leq x \leq \frac{\pi}{3}$ $hx = \frac{\pi}{24}$	*Массив* Y. *Сумма и количество элементов массива* Y, *лежащих на отрезке* [0;2].

Продолжение таблицы к заданию 5

1	2	3	4
12	$z_i = \begin{cases} \sqrt{\dfrac{\lvert x_i \rvert}{1+x_i^2}}, & \text{если} \quad x_i \le 2 \\ \sqrt[3]{x_i^2}+1, & \text{если} \quad x_i > 2 \end{cases}$ *Элементы* Z_i *сгладить по формуле:* $Z_i = (Z_{i-1} + Z_i + Z_{i+1})/3$	*Массив* X $i = 1 \div 15$	*Массив* Z *до и после сглаживания.*
13	$a_i = \begin{cases} x_i^2 + 2x_i - 5, & \text{если } x_i < 0 \\ 2x_i + \cos \pi / x_i, & \text{если } x_i \ge 0 \end{cases}$ $S = \sum_{a_i>0} a_i, \quad P = \prod_{a_i<0} a_i$	*Массив* X $-5 \le x_i \le 4$ $hx_i = 0.9$	*Массивы* X, A. *S, P.* *Количество* $a_i < 0$.
14	$z_i = \begin{cases} x_i^2 + \dfrac{\pi}{2}\sin\dfrac{\pi}{2}x_i, & \text{если } y_i \ge 1 \\ 1+\sqrt{\lvert x_i \rvert}, & \text{если } y_i < 1 \end{cases}$ $y_i = x_i^2 - 2/x_i$ *Считать пары точек* (y_i, z_i) *координатами точек на плоскости* YOZ	*Массив* X $i = 1 \div 10$	*Массивы* Y, Z. *Определить, какая из точек* 2, 3…10 *наиболее удалена от точки* (y_1, z_1).
15	$y_i = \begin{cases} \sin a_i^2 + \cos(a_i - \pi), \text{если } a_i \ge \pi \\ \dfrac{a_i^2 + a_i - 3}{a_i + \sqrt{a_i^2+1}}, \quad \text{если } a_i < \pi \end{cases}$ *Заменить все отрицательные элементы массива* Y *суммой R и значения соответствующего элемента.*	*Массив* A $i = 1 \div 11$	*Массив* Y *до и после замены. Ср. арифметическое (R) элементов массива* Y.
16	$y_i = \dfrac{\pi}{2}\sin\dfrac{\pi}{2}x_i - 0.5\cos\dfrac{x_i}{3}$ $\upsilon_i = \begin{cases} y_i + x_i\sqrt{1+0.5\sin x_i}, \text{если } y_i > 0.5 \\ 3\ln(1+e^{y_i}), \text{ если} \quad y_i \le 0.5 \end{cases}$ *Считать* (V_i, Y_i) *координатами точек плоскости. Определить процент (PR) точек, лежащих в круге радиусом R с центром в точке*(V_0, Y_0).	*R,* V_0, Y_0, *Массив* X $i = 1 \div 10$	*R,* V_0, Y_0, *PR.* *Массивы* Y, V.

Продолжение таблицы к заданию 5

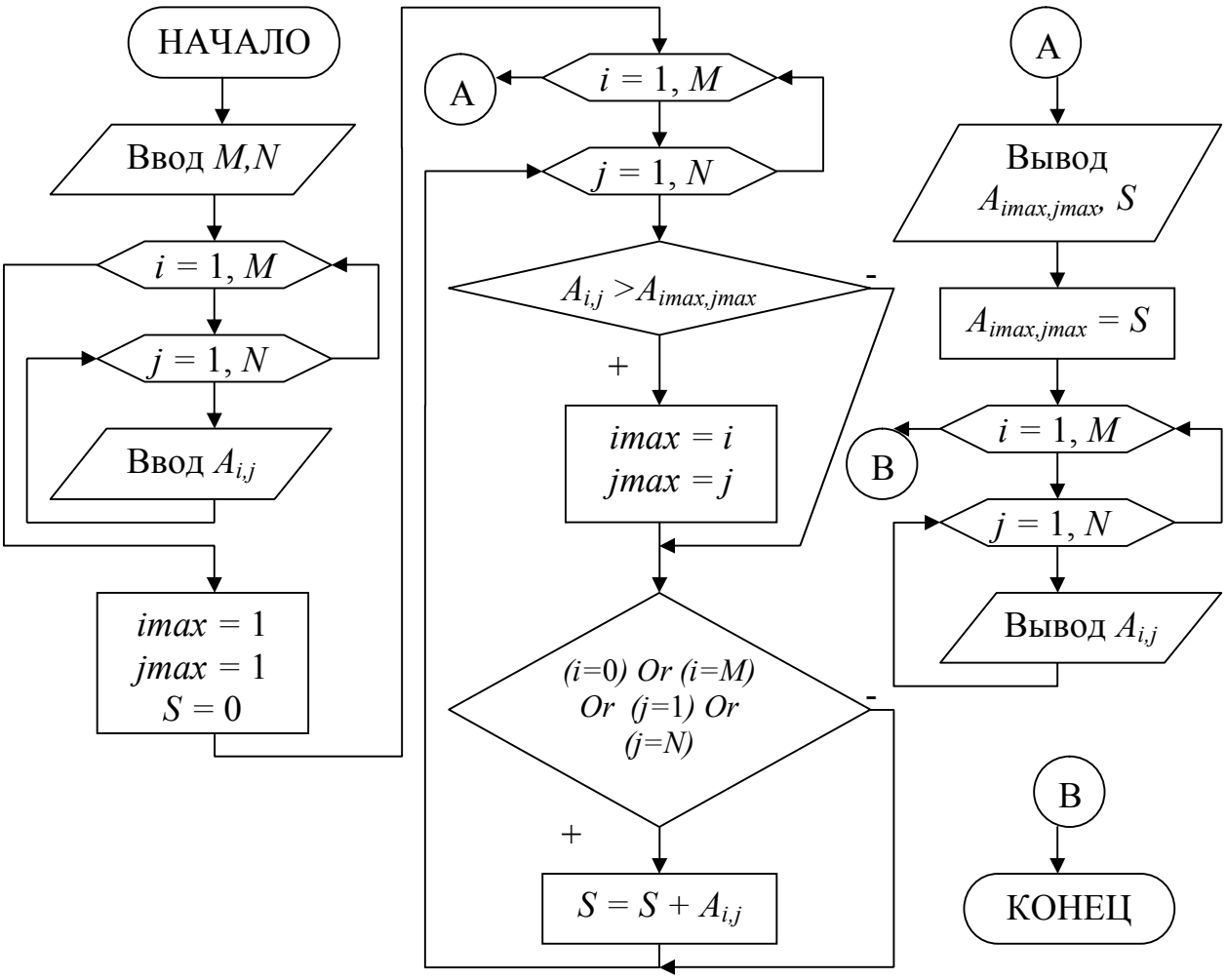
1	2	3	4
17	$b_i = \sqrt[4]{a_i^2+1} + \sqrt[3]{a_i^2+1}$ $c_i = \begin{cases} 2e^{0.5a_i}, \text{ если } \lvert a_i\rvert < 5 \\ 2\pi \sin \pi a_i + a_i, \text{ если } \lvert a_i\rvert \ge 5 \end{cases}$ *Считать a_i, b_i, c_i коэффициентами квадратного уравнения* $ax^2+bx+c=0$	*Массив* A i=1÷20	*Массивы* B, C. *Порядковые номера уравнений, имеющих комплексные корни.*
18	$a_i = \begin{cases} \dfrac{1}{\sqrt[3]{x_i}(e^{0.1x_i}+1.5)}, \text{ если } x_i \ge 0.5 \\ 1.8\sqrt{\lvert x_i\rvert+1} + e^{0.1x_i}, \text{если } x_i < 0.5 \end{cases}$ $b_i = \sin \pi a_i$ *Считать значения элементов массива* A *и* B *длинами полуосей эллипса a и b.*	*Массив* X $4 \le x_i \le 12$ hx_i=2	*Массивы* A, B. *Порядковый номер N эллипса, площадь которого* $S=\pi \cdot a \cdot b$ *наибольшая.*
19	$y_i = \begin{cases} \ln \dfrac{1}{2+2x_i+x_i^2}, \text{если} -1.5 \le x_i \le 0 \\ arctg\ x_i, \text{ если } x_i > 0 \\ x_i^2, \text{ если } x_i < -1.5 \end{cases}$	*Массив* X i=1÷12	*Массив* Y. *Индекс элемента, наиболее близкого по значению к ср. геометрическому (P) массива* Y.
20	$y_i = \begin{cases} 4x_i^{0.6} - 2\sqrt{x_i}, \text{ если } 1 \le x_i \le 10 \\ 0.5x_i + 1, \text{ если } x_i > 10 \\ 100x_i^2 - 5e^{x_i}, \text{ если } x_i < 1 \end{cases}$	*Массив* X -0.3≤ x_i ≤1.2 hx_i=0.3	*Массив* Y. *Среднее арифметическое (A) массива* Y *и количество* $y_i > A$.

Задание № 6. Обработка двумерных массивов

Задание. Составить блок-схему алгоритма и программу решения поставленной задачи обработки двумерного массива.

Пример. Заменить максимальный элемент матрицы A размерностью *M*x*N* суммой элементов, расположенных на периметре матрицы.

Блок-схема алгоритма:



Программа на языке Турбо Паскаль:

```
Program Pr6;
Var a: array [1..10, 1..10] of real;
      i, j, M, N, imax, jmax: byte;
      S: real;
Begin
 Write('Ввод M, N'); Readln(M,N);
 for i:=1 to M do
  for j:=1 to N do begin
  Write('Ввод A[', i, j, '] ='); Readln(a[i,j]);
 end;
 imax := 1; jmax: = 1;
 S: = 0;
 for i:=1 to M do
  for j:=1 to N do begin
   if a[i,j] >= a[imax,jmax] then begin
```

```
    imax := i;   jmax := j; end;
 if (i=0) Or (i=M) Or  (j=1) Or (j=N) then
          S := S + a[i,j]
end;
writeln('A[', imax, jmax, ']=', a[imax, jmax] : 6 : 2, '  S=', S : 6 : 2);
a[imax, jmax] : = S;
for i:=1 to M do begin
  for j:=1 to N do
  Write('A[', i, j, '] =', a[i,j] : 6 : 2, '  ' );
  writeln  end;
End.
```

Варианты заданий.

1. Определить номера строки и столбца максимального отрицательного элемента прямоугольной матрицы A= $(a_{i,j})_{M,N.}$

2. Найти среднее арифметическое значение элементов прямоугольной матрицы X=$(x_{i,j})_{7,3}$, находящихся на периметре этой матрицы.

3. Сформировать вектор D=$(d_1,d_{2,}...d_M)$, каждый элемент которого равен среднему арифметическому значений элементов строк матрицы C размерностью $M\times N$.

4. В матрице A= $(a_{i,j})_{M,M}$ элементы главной диагонали заменить «1», если данный элемент больше последующих элементов соответствующей строки, и «0» - в противном случае.

5. Вычислить элементы вектора G=$(g_1,g_2,...g_M)$, как произведение отрицательных элементов соответствующих строк заданной матрицы A размерностью $M\times N$.

6. Рассчитать элементы матрицы C=$(c_{i,j})_{3,3}$, являющейся произведением матриц A=$(a_{i,j})_{3,4}$ и B=$(b_{i,j})_{4,3}$. Элементы матрицы C рассчитываются по формуле:

$$C_{ij} = \sum_{l=1}^{n} a_{il} \cdot b_{lj}$$

7. Подсчитать количество нулевых элементов матрицы размерностью $M\times N$ и напечатать их индексы. Первый по счету нулевой элемент заменить суммой положительных элементов.

8. Вычислить элементы матрицы Z размерностью 5×6 по элементам исходной матрицы X=$(x_{i,j})_{5,6}$. Главную диагональ оставить неизменной. $z_{i,j}=x^2_{i,j.}$

9. Сформировать вектор B=$(b_1,b_2...b_7)$, каждый элемент которого определяется как минимальный элемент соответствующего столбца исходной матрицы

A=$(a_{i,j})_{6,7.}$

10. Преобразовать исходную матрицу A=$(a_{i,j})_{5,7}$ так, чтобы последний элемент каждой строки был заменен суммой предыдущих элементов той же строки.

11. Преобразовать заданную матрицу B=$(b_{i,j})_{4,6}$ таким образом, чтобы первый элемент каждого столбца был заменен произведением последующих элементов того же столбца.

12. Преобразовать матрицу C=$(c_{i,j})_{8,8}$ так, чтобы все элементы расположенные ниже главной диагонали, были уменьшены вдвое, а элементы, расположенные выше главной диагонали – увеличены вдвое.

13. Определить количество и номера отрицательных элементов в матрице A размерностью 6×7. Последний по счету отрицательный элемент заменить суммой положительных элементов матрицы.

14. Найти отношение минимального элемента матрицы A=$(a_{i,j})_{5,6}$ к максимальному элементу матрицы B=$(b_{i,j})_{7,8}$.

15. В заданной матрице A=$(a_{i,j})_{4,5}$ найти нулевой элемент с наибольшим значением индекса i и все элементы столбца, в котором находится этот элемент, обнулить. Если в матрице нет нулевых элементов, отпечатать соответствующее сообщение.

16. Найти отношение количества положительных элементов к количеству элементов отрицательных заданной матрицы F. В случае, если матрица F не содержит отрицательных элементов, то все положительные элементы уменьшить на 1.

17. Для квадратной матрицы F=$(f_{i,j})_{7,7}$ найти отношение суммы элементов, расположенных ниже главной диагонали, к сумме элементов, расположенных выше главной диагонали, предусмотрев соответствующее сообщение, если последняя сумма (делитель) окажется равной 0.

18. В заданной матрице B=$(b_{ij})_{6,7.}$ Найти элемент $b_{ij}<5$ с наибольшим значением индекса j. Все элементы столбца, в котором находится искомый элемент (кроме него) сделать равными 1.

19. Получить матрицу-строку B, каждый элемент которой равен среднему геометрическому значений элементов соответствующей строки матрицы A размерностью 9×10.

20. Все элементы матрицы C=$(c_{ij})_{9,9}$, расположенные выше главной диагонали преобразовать, умножив их на минимальный элемент матрицы C.

Задание № 7. Использование процедур и функций

Задание. Составить блок-схему алгоритма и программу решения поставленной задачи, которая в соответствии с исходными данными вычисляет значения заданных выражений. Расчет элементов массива и подпрограмму оформить в виде процедур с параметрами.

Варианты заданий.

1. Для одномерного массива чисел $A_1,A_2,\ldots A_m$ составить подпрограмму определения значения математического ожидания по формуле: $SM = \frac{1}{m}\sum_{k=1}^{m} A_k$. Подпрограмму использовать для определения значения математического ожидания массивов: C[60], R[80], S[100] и U[70], элементы которых вычисляются по формулам: $C_i=2.8+5.1\cdot sin(i/2+0.5)$, $R_j=6.6+4.7\cdot sin(j/3-1.2)$,
$S_l=1.7-3.6\cdot sin(l/4+0.2)$, $U_n=7.8+15.6\cdot sin(n/2+0.9)$.

2. Составить подпрограмму подсчета величины процента положительных элементов – P в одномерном массиве $A_1,A_2,\ldots A_m$. Использовать подпрограмму для обработки массивов D[70], F[80], R[40], P[30], элементы которых вычисляются по формулам: $D_i=3.1i^2-59.7i-16.2$, $F_j=5.4j^2-82.1j+29.6$,
$R_k=-6.9k^2-+31.2k-10.5$, $P_m=-11.5m^2+48.2m+15.6$.

3. Составить подпрограмму определения разности g между максимальным и минимальным значениями элементов одномерного массива b_1, $b_2,\ldots b_m$. Подпрограмму использовать для обработки массивов A[40], D[80], H[50] и Q[70], элементы которых вычисляются по формулам:
$A_i=-(i-2)^2+i^3$, $D_j=1.5(j-3.2)^2-1.1j^3$,
$H_k=2.7(k-0.2)^2-0.5k^3$, $Q_n=3.3(n+2.5)^2-1.8n^3$.

4. Составить подпрограмму определения номера строки матрицы A(M,N), в которой находится наибольший по абсолютной величине элемент матрицы. Подпрограмму применить для обработки матриц: X[30,40], Y[50,50], Z[60,40], элементы которых вычисляются по формулам:
$X_{ij}=5.7\cdot j\cdot sin(i/2)+9.3\cdot i\cdot cos(j/2)$, $Y_{ij}=12.1\cdot j\cdot sin(i/2)-3.8\cdot i\cdot cos(j/2)$,
$Z_{ij}=10.5\cdot j\cdot sin(i/2)+23.4\cdot i\cdot cos(j/2)$

5. Составить подпрограмму формирования массива $D=[d_1,d_2,d_3,d_4]$, где d_i – максимальный по модулю элемент соответственно массивов A[140], B[80], C[90], F[70], элементы которых вычисляются по формулам:

$A_i=9.6i-15.3\cdot tg(i^2-0.5);$ $B_j=11.6j-18.3\cdot tg(j^2+1.5);$

$C_k=-11.2k+10.1\cdot tg(k^2-3.9);$ $F_l=19.6\ l-29.4\cdot tg(l^2-3.3).$

6. Составить подпрограмму формирования массива $S=[S_1,S_2,S_3,S_4]$, где S_i – минимальный положительный элемент массива A[12], B[16], C[20], D[8], соответственно. Элементы массивов вычисляются по формулам:

$A_i=3.8i^2-12.4i+5.1,\ B_i=5.6i^2+11.5i-29.3,\ C_k=18.1k^2-6.8k-9.9,\ D_l=10.5l^2-21.6l+6.9.$

7. Составить подпрограмму формирования вектора $Q=[Q_1,Q_2,Q_3,Q_4]$, компоненты которого равны произведению ненулевых элементов массивов A[9], F[10], Z[6], D[7], элементы которых вычисляются по формулам:

$A_i=1.2\cdot (i-2)\cdot\sin(i),\ F_j=5.9\cdot (j-5)\cdot\sin(j),\ Z_k=12.3\cdot (k-4)\cdot\sin(k),\ D_m=8.6\cdot (m-1)\cdot\sin(m).$

8. Составить подпрограмму вычисления скалярных произведений векторов X и У по формуле $S=\sum_{i=1}^{N} x_i y_i$. Подпрограмму использовать для определения скалярных произведений векторов: B и C, C и Z, Z и S, B и Z. Координаты векторов определить из выражений:

$B_i=2.8-(i+4.5)^2,\ C_i=-12.6+(i-2.2)^2,\ Z_i=-9.1-(i+5.7)^2,\ S_i=8.5-(i-2.7)^2,$ *где* $i=1\ldots100.$

9. Составить подпрограмму нахождения разности между суммой элементов с четными индексами и суммой элементов с нечетными индексами в массивах C[60], R[80], S[100], U[70], элементы которых вычисляются по формулам:

$C_i=2.8+5.1\cdot\sin(i/2+0.5),$ $R_j=6.6+4.7\cdot\sin(j/3-1.2),$

$S_l=1.7-3.6\cdot\sin(l/4+0.2),$ $U_n=7.8+15.6\cdot\sin(n/2+0.9).$

10. Составить подпрограмму нахождения суммы положительных элементов одномерного массива $x_1,x_2,\ldots x_n$. Подпрограмму использовать для обработки массивов A[90], F[100], Z[60], P[70], элементы которых вычисляются по формулам:

$A_i=1.2\cdot (i-2)\cdot\sin(i),\ F_j=5.9\cdot (j-5)\cdot\sin(j),\ Z_k=12.3\cdot (k-4)\cdot\sin(k),\ P_m=8.6\cdot (m-1)\cdot\sin(m).$

11. Составить подпрограмму, которая по исходным массивам C[60], R[80], S[100] формирует вектор $Q=[Q_1,Q_2,Q_3]$, где компоненты Q_i равны произведению отрицательных значений элементов массива. Элементы массивов вычислять по формулам: $C_i=2.8+5.1\cdot\sin(i/2+0.5),\ R_j=6.6+4.7\cdot\sin(j/3-1.2),\ S_l=1.7-3.6\cdot\sin(l/4+0.2).$

12. По исходным массивам A[40], D[80], H[50] сформировать массив $G=[G_1,G_2,G_3]$, компоненты которого равны суммам абсолютных значений исходных массивов. Расчет значения суммы оформить в виде отдельной программы. Элементы массивов вычислять по формулам:

$A_i=-(i-2)^2+i^3,\ D_j=1.5(j-3.2)^2-1.1j^3,\ H_k=2.7(k-0.2)^2-0.5k^3.$

13. Заданы матрицы C_{ij}, D_{jm}, S_{mi} (где i=1...40, j=1…30, m=1…50). Составить подпрограмму определения суммы элементов k строк каждой матрицы. Значение k задать при вводе. Элементы массивов определять по формулам:

$C_{ij}=(i-3.5)\cdot(j+1.7)$, $D_{jm}=(j+4.2)\cdot(m-5.6)$, $S_{mi}=(m-7.6)\cdot(i+5.2)$.

14. Составить подпрограмму для расчета общего количества элементов массивов A[140], B[80], C[90], которые принадлежат отрезку [m;n]. Значения m и n задаются при вводе. Элементы массивов определять по формулам:

$A_i=9.6i-15.3\cdot tg(i^2-0.5)$; $B_j=11.6j-18.3\cdot tg(j^2+1.5)$; $C_k=-11.2k+10.1\cdot tg(k^2-3.9)$.

15. Составить подпрограмму, которая по исходным массивам A[20], B[180], C[60], D[30] формирует массив Q=[Q_1,Q_2,Q_3,Q_4], компоненты которого равны минимальным по абсолютной величине элементам массивов A,B,C,D. Элементы исходных массивов вычислять по формулам:

$A_i=14.4i-2.9e^{\sin(i)}$, $B_j=-8.5j+1.6e^{\sin(j)}$; $C_k=11.3k-4.7e^{\sin(k)}$, $D_l=-18.1e+12.9e^{\sin(l)}$.

16. Составить подпрограмму для определения суммы элементов квадратной матрицы X[n,n], лежащих на главной диагонали, и использовать ее для обработки матриц A[10,10], B[40,40], C[80,80]. Элементы матриц определяются по формулам: $A_{ij}=3.7-8.2ij^2+10.4i^2j$, $B_{ij}=-5.2+13.9ij^2-4.6i^2j$, $C_{ij}=8.4+4.6ij^2-7.5i^2j$.

17. Составить подпрограмму подсчета произведения положительных элементов массива X и использовать ее для обработки массивов A[12], B[16], C[20], D[8], элементы которых вычислять по формулам:

$A_i=3.8i^2-12.4i+5.1$, $B_i=5.6i^2+11.5i-29.3$, $C_k=18.1k^2-6.8k-9.9$, $D_l=10.5l^2-21.6l+6.9$.

18. Составить подпрограмму определения отношения максимального и минимального элемента массива X(n), с помощью которой рассчитать z=a+b+c, где a,b,c – отношения максимальных и минимальных элементов массивов A[140], B[80], C[90], F[70], рассчитываемых по формулам:

$A_i=9.6i-15.3\cdot tg(i^2-0.5)$; $B_j=11.6j-18.3\cdot tg(j^2+1.5)$;

$C_k=-11.2k+10.1\cdot tg(k^2-3.9)$; $F_l=19.6\,l-29.4\cdot tg(l^2-3.3)$.

19. Заданы массивы A[12], B[16], C[20], D[8]. Составить подпрограмму для нахождения разницы между произведениями элементов с четными индексами и произведениями элементов с нечетными индексами каждого из массивов. Элементы массивов рассчитать по формулам:

$A_i=3.8i^2-12.4i+5.1$, $B_i=5.6i^2+11.5i-29.3$, $C_k=18.1k^2-6.8k-9.9$, $D_l=10.5l^2-21.6l+6.9$.

20. Составить подпрограмму определения суммы элементов квадратной матрицы, лежащих выше главной диагонали, которую использовать для обработки

матриц X[30,30], Y[50,50], Z[60,60]. Элементы указанных матриц определить по формулам:

$X_{ij}=5.7\cdot j\cdot sin(i/2)+9.3\cdot i\cdot cos(j/2)$, $Y_{ij}=12.1\cdot j\cdot sin(i/2)-3.8\cdot i\cdot cos(j/2)$,

$Z_{ij}=10.5\cdot j\cdot sin(i/2)+23.4\cdot i\cdot cos(j/2)$.

Рекомендации к выполнению контрольной работы

Для выполнения контрольной работы составить блок-схему алгоритма решения задачи и программу на языке программирования Турбо Паскаль в соответствии с выбранным вариантом задания.

Номер варианта задания выбирается по буквам фамилии студента в соответствии с таблицей.

№ варианта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Буквы	А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф
	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я									

Например, для студента с фамилией Сидоров варианты в каждом задании будут такими:

1 задание – 17 вариант (буква С)

2 задание – 9 вариант (буква И)

3 задание – 5 вариант (буква Д)

4 задание – 14 вариант (буква О)

5 задание – 16 вариант (буква Р)

6 задание – 14 вариант (буква О)

7 задание – 3 вариант (буква В)

Если фамилия состоит меньше, чем из 7 букв, то в качестве недостающих взять первые буквы имени.

МЕТОДИЧЕСКОЕ ПОСОБИЕ

«ОСНОВЫ АЛГОРИТМИЗАЦИИ»

(для студентов заочной формы обучения)

Составители: Ефименко Константин Николаевич
Добровольский Юрий Николаевич
Ильяшенко Владимир Степанович

Подп. в печать 12.09.06 г. Формат 60x84 1/16. Бумага KumLux.
Ризографическая печать. Усл. печ. л. 4,65 Усл. кр.-отт. 4,70
Уч.-изд. л. 4,75 Тираж 50 экз. Заказ № 20/10

Донецкий национальный технический университет
83000, г. Донецк, ул. Артёма, 58