

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Комсомольский-на-Амуре государственный технический университет»

М.Е. Щелкунова

ИНФОРМАТИКА

Комсомольск-на-Амуре 2012

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
1. ОСНОВЫ АЛГОРИТМИЗАЦИИ.....	4
1.1. ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА ПЭВМ.....	4
1.2. АЛГОРИТМ. СВОЙСТВА АЛГОРИТМОВ	4
1.3. СПОСОБЫ ЗАПИСИ АЛГОРИТМОВ.....	5
1.4. ИЗОБРАЖЕНИЕ АЛГОРИТМА В ВИДЕ БЛОК-СХЕМЫ	6
1.5. РЕКОМЕНДАЦИИ ПО РАЗРАБОТКЕ АЛГОРИТМОВ	9
1.6. СТРУКТУРЫ АЛГОРИТМОВ И СПОСОБЫ ИХ ЗАПИСИ В ВИДЕ БЛОК-СХЕМЫ	10
1.6.1. АЛГОРИТМЫ ЛИНЕЙНОЙ СТРУКТУРЫ (СЛЕДОВАНИЕ)	10
1.6.2. АЛГОРИТМЫ РАЗВЕТВЛЕННОЙ СТРУКТУРЫ (ВЕТВЛЕНИЕ)	12
1.6.3. АЛГОРИТМЫ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ (ПОВТОРЕНИЕ)	19
1.7. РАБОТА С МАССИВАМИ.....	38
1.7.1. ПОНЯТИЕ МАССИВА.....	38
1.7.2. АЛГОРИТМЫ РАБОТЫ С ОДНОМЕРНЫМИ МАССИВАМИ	42
1.7.3. АЛГОРИТМЫ СОРТИРОВКИ ОДНОМЕРНЫХ МАССИВОВ	58
1.7.4. АЛГОРИТМЫ РАБОТЫ С ДВУМЕРНЫМИ МАССИВАМИ	63
РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ.....	74
ЗАДАНИЕ 1.....	74
ЗАДАНИЕ 2.....	75
ЗАДАНИЕ 3.....	77
ЗАДАНИЕ 4.....	80
ЗАДАНИЕ 5.....	81
ЗАДАНИЕ 6.....	84
ЗАДАНИЕ 7.....	86
ЗАДАНИЕ 8.....	89
ЗАДАНИЕ 9.....	91
ЗАДАНИЕ 10.....	92
ЗАДАНИЕ 11.....	94
ЗАДАНИЕ 12.....	95
ЗАДАНИЕ 13.....	97
ЗАДАНИЕ 14.....	98
ЛИТЕРАТУРА.....	100

ВВЕДЕНИЕ

Дисциплина «Информатика» изучается в рамках цикла естественно-научных дисциплин (федеральная компонента) учебных планов подготовки бакалавров техники и технологии направления 230100 «Информатика и вычислительная техника» и бакалавров информационных систем направления 230400 «Информационные системы и технологии» и является первой в ряду дисциплин, нацеленных на системное изучение информационных технологий для получения доступа и обработки любых видов информации посредством компьютера и линий связи. Знания, умения и навыки, полученные вами в ходе обучения дисциплине «Информатика», могут быть использованы при изучении практически всех обще-профессиональных и специальных дисциплин.

При изучении дисциплины «Информатика»:

- вы получите прочные теоретические знания по данной дисциплине;
- приобретёте практические навыки разработки алгоритмов решения задач и представления их в виде блок-схем;
- ...

Занятия по информатике построены классическим образом и включают в себя три компонента: лекционные занятия, лабораторные занятия, самостоятельную подготовку в виде выполнения расчетно-графических заданий.

Особенность занятий по информатике в том, что предполагается большое количество часов самостоятельного увлеченного изучения дисциплины за столом с листом бумаги и карандашом в руках для приобретения навыков разработки алгоритмов решения задач и представления их в виде блок-схем.

Итак, порядок изучения дисциплины «Информатика» следующий:

1) имея большое искреннее желание научиться разрабатывать алгоритмы, вы, внимательно читая главу 1 и увлеченно разбирая с карандашом в руке приведенные в ней примеры, самостоятельно изучаете основы построения алгоритмов и представления их в виде блок-схем;

2) после этого, опираясь на приобретенный опыт и примеры раздела 1, легко можете выполнить и выполняете, оформляете, сдаёте на проверку и защищаете предлагаемое вам расчетно-графическое задание.

1. ОСНОВЫ АЛГОРИТМИЗАЦИИ

1.1. ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА ПЭВМ

Решение задачи разбивается на следующие этапы:

1) **постановка задачи.** Постановка задачи – это формулировка задачи, излагаемая в терминах некоторой конкретной области науки, техники, медицины, экономики и так далее. Она должна содержать все необходимые для решения задачи сведения. При постановке задачи выясняется конечная цель и вырабатывается общий подход к решению задачи; выясняется, сколько решений имеет задача и имеет ли их вообще; изучаются общие свойства рассматриваемого физического явления или объекта; анализируются возможности данной системы программирования;

2) **формализация (математическая постановка).** На этапе математической постановки задачи поставленную задачу формулируют, как задачу некоторого раздела математики, то есть задачу **формализуют**. Это означает, что, построив математическую модель, можно отказаться от рассмотрения и анализа реального объекта и перейти к анализу математической модели, выявлению исходных данных и результатов, как параметров математической модели. **Математическая модель – это математические соотношения, отражающие суть реального объекта**, учитывающие существенные для решаемой задачи параметры объекта и связывающие исходные данные с результатом. Получая математические соотношения, надо стремиться к тому, чтобы математическая задача, к которой сводится решаемая задача, была типовой, известной задачей математики. Итак, на этапе формализации все объекты задачи описываются на языке математики, выбирается форма хранения данных, составляются все необходимые формулы;

3) **выбор (или разработка) метода решения.** Выбор существующего или разработка нового метода решения – очень важный и, в то же время творческий этап;

4) **разработка алгоритма.** На этом этапе метод решения записывается применительно к данной задаче на одном из алгоритмических языков (чаще на графическом языке);

5) **составление программы.** Перевод решения задачи на язык, понятный машине;

6) **отладка программы.** Поиск и исправление ошибок;

7) **вычисление и обработка результатов.**

1.2. АЛГОРИТМ. СВОЙСТВА АЛГОРИТМОВ

Алгоритм – четкое описание последовательности действий, которые необходимо выполнить при решении задачи. Можно сказать, что алгоритм описывает процесс преобразования исходных данных в результаты, так как для решения любой задачи необходимо:

- ввести исходные данные;
- преобразовать исходные данные в результаты (выходные данные);
- вывести результаты.

Алгоритм – это определенным образом организованная последовательность действий, за конечное число шагов приводящая к решению задачи.

Алгоритмами, например, являются правила сложения, умножения, решения алгебраических уравнений, умножения матриц и т.п.

Слово алгоритм происходит от **algoritmi**, являющегося латинской транслитерацией арабского имени хорезмийского математика IX века **аль-Хорезми**. Благодаря

латинскому переводу трактата аль-Хорезми европейцы в XII веке познакомились с позиционной системой счисления, и в средневековой Европе алгоритмом называлась десятичная позиционная система счисления и правила счета в ней.

Алгоритм – это точная **инструкция**, а инструкции встречаются практически во всех областях человеческой деятельности. Возможны алгоритмы проведения физического эксперимента, сборки шкафа или телевизора, обработки детали. Однако не всякая инструкция есть алгоритм. Инструкция становится алгоритмом только тогда, когда она удовлетворяет определенным *требованиям*.

Основными свойствами алгоритмов являются:

- **понятность**. Алгоритм рассчитан на конкретного исполнителя и должен включать только те команды, которые этому исполнителю доступны, которые входят в его систему команд;
- **дискретность**, т.е. последовательное выполнение простых шагов или ранее определённых (подпрограммы) шагов. Преобразование исходных данных в результат осуществляется дискретно во времени;
- **определенность (однозначность, детерминированность)**. Состоит в совпадении получаемых результатов независимо от пользователя и применяемых технических средств (однозначность толкования инструкций). При одних и тех же данных получается один и тот же результат;
- **завершаемость (конечность)**. При корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов. С другой стороны, вероятностный алгоритм может и никогда не выдать результат, но вероятность этого равна **НУЛЮ**;
- **результативность**. Означает возможность получения результата после выполнения конечного количества операций;
- **массовость (универсальность)**. Заключается в возможности применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных (разработка в общем виде).

1.3. СПОСОБЫ ЗАПИСИ АЛГОРИТМОВ

Разработка алгоритма решения задачи – это разбиение задачи на последовательно выполняемые этапы, причем результаты выполнения предыдущих этапов могут использоваться при выполнении последующих. При этом должны быть четко указаны как содержание каждого этапа, так и порядок выполнения этапов.

Отдельный этап алгоритма должен быть достаточно простым и понятным без пояснений, либо должен представлять собой другую, более простую задачу, алгоритм решения которой известен (разработан заранее).

Разработанный алгоритм можно записать несколькими **способами**:

- на естественном языке (словесный способ);
- на языке формул (словесно-формульный способ);
- в виде таблицы;
- в виде граф-схемы (граф – совокупность точек (называются вершинами) и линий (называются рёбрами), в которой каждая линия соединяет две точки);
- в виде сетей Петри;
- в виде блок-схемы (структурный способ);
- на алгоритмическом языке (в виде программ).

Словесная запись алгоритма рассчитана на исполнителя – человека. Примерами алгоритмов на естественном языке являются четыре правила арифметики для сложения, вы-

числения, умножения и деления чисел – они записываются словами. Другими примерами алгоритмов на естественном языке являются правила геометрических построений.

Словесно-формульный способ используется, например, для записи правила вычисления корней квадратного уравнения $ax^2+bx+c=0$ на множестве действительных чисел:

- 1) ввести в компьютер числовые значения переменных **a**, **b** и **c**;
- 2) вычислить **d** по формуле $d=b^2-4ac$;
- 3) если $d<0$, то напечатать сообщение «Корней нет» и перейти к пункту 4. Иначе вычислить и напечатать значения **x**₁ и **x**₂;
- 4) прекратить вычисления.

Табличный способ хорошо подходит для записи алгоритма вычисления значений нескольких функций **y**, **y**₁, **y**₂ от значения **x**. Составляется следующая таблица:

x	$y = x^3$	$y_1 = 2 \cdot x^3$	$y_2 = 3 \cdot x^3 + x$
0,25			

Табличный способ применяется и при решении задачи табулирования функции на заданном отрезке. Составляется следующая таблица:

x	0,25	0,35	0,45	0,55	0,65
$y = x^3$					

На компьютере алгоритмы можно реализовать *в виде программ*. Можно сказать, что программа, это алгоритм, записанный на языке, понятном ПЭВМ.

Перед составлением программ чаще всего используются словесно-формульный и блок-схемный способы.

Какой бы способ записи алгоритма не использовался важно, чтобы каждый этап вычислений, каждая команда алгоритма были понятны исполнителю, точно определяли все его действия и могли бы быть им выполнены.

Для решения одной и той же задачи можно разработать несколько алгоритмов. Каждый алгоритм создается в расчете на вполне конкретного исполнителя. Если планируется реализация алгоритма на машине, следует учитывать его приспособляемость к ПЭВМ, простоту и время реализации на машине.

Алгоритм должен быть разработан таким образом, чтобы исполнитель мог даже *не вникать в смысл* того, что он делает, и вместе с тем *получать нужные результаты*. Исполнение уже имеющегося алгоритма можно поручить субъекту или объекту, который не обязан вникать в существо дела, а возможно, и не способен его понять.

1.4. ИЗОБРАЖЕНИЕ АЛГОРИТМА В ВИДЕ БЛОК-СХЕМЫ

Блок-схема – это наглядное графическое изображение структуры алгоритма, в котором каждый этап вычислений изображается какой-либо геометрической фигурой (блоком), внутри которой дается описание соответствующего действия. Блоки соединяются между собой линиями. Алгоритм выполняется в соответствии с расположением блоков и их соединением.

Графический метод описания алгоритма очень *наглядный*, но в случае сложного алгоритма схема получается громоздкой, и это ее достоинство теряется. *Использование схем алгоритмов значительно упрощает написание программ начинающими программистами.*

В настоящее время действует стандарт **ГОСТ 19.701-90 (ИСО 5807-85)** «Схема алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Утвержден стандарт 1 января 1992 года.

Наиболее распространенные обозначения блоков, которые используются в схемах алгоритмов, приведены в таблице 1.1.

Таблица 1.1

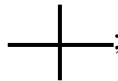
Символы блок-схем

Символ	Наименование символа	Назначение символа
	Данные	Обозначение действий по вводу, выводу данных. Надпись на блоке: слово ВВОД/ВЫВОД или ПЕЧАТЬ и список вводимых/выводимых переменных
	Процесс (арифметический блок)	Отображает функцию обработки данных любого вида (операцию присваивания). Надпись на блоке: операция или группа операций
	Предопределенный процесс	Используется для вызова подпрограмм (библиотечных или пользовательских). Надпись на блоке: название подпрограммы с указанием входных и выходных параметров
	Решение (условный блок)	Используется для обозначения выбора направления движения по схеме. Надпись на блоке: условие. В результате проверки условия осуществляется выбор одного из возможных путей (ветвей) вычислительного процесса. Если условие выполняется, то следующим выполняется этап по ветви + / ДА , если условие не выполняется, то выполняется этап по ветви - / НЕТ
	Граница цикла	Используются для обозначения начала и конца цикла. Надпись на блоке: начальное, конечное значения и правило изменения значения переменной, являющейся счетчиком цикла
	Терминатор	Используется для обозначения начала и конца схемы. Надпись на блоке: НАЧАЛО/ КОНЕЦ
	Соединитель	Используется для указания связи между прерванными линиями потока, связывающими блоки. Надпись на блоке: любая буква или цифра; каждая пара соединителей обозначается уникальными надписями
	Комментарий	Используется для пояснений к отдельным блокам

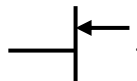
	Линия	Используется для соединения блоков
--	-------	------------------------------------

Некоторые *правила* для *построения схем*:

- естественным направлением линии считается сверху – вниз и слева – направо. В этом случае стрелки не ставятся. В остальных случаях ставятся;
- необходимо избегать пересечения линий, например, таких:



- если две или более линий объединяются в одну линию, то место объединения должно быть смещено, например, так:



Пример 1

Решить квадратное уравнение.

Дано (входные данные): **a, b, c** – коэффициенты квадратного уравнения.

Найти (выходные данные): **x1, x2** – корни квадратного уравнения.

В качестве примера рассмотрим блок-схему алгоритма решения квадратного уравнения $a \cdot x^2 + b \cdot x + c = 0$ на множестве действительных чисел.

Данный алгоритм (рис. 1.1) рассчитан на два случая решения: либо существуют два действительных различных дискриминанта корня (дискриминант $d > 0$), либо действительные корни отсутствуют (дискриминант $d < 0$), и не учитывает случая существования единственного корня ($a = 0, b \neq 0$).

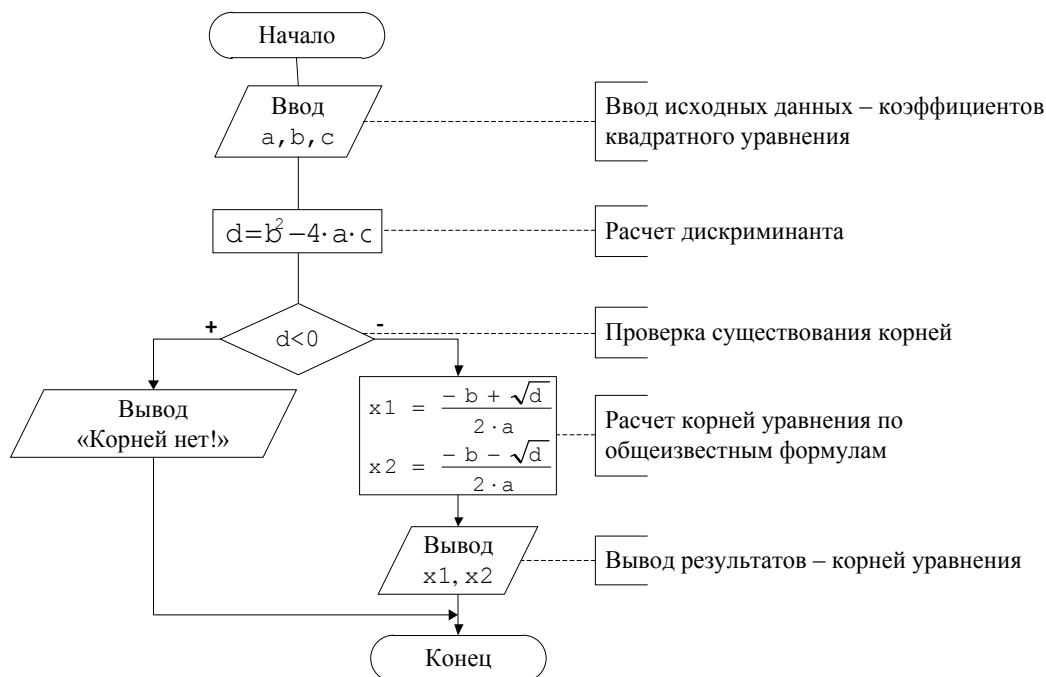


Рис. 1.1. Блок-схема алгоритма решения квадратного уравнения

Для нахождения корней воспользуемся общеизвестными формулами.

Знак '=' означает не математическое равенство, а **операцию присваивания**. Переменной, стоящей слева от оператора, присваивается значение, указанное справа. Причем это значение может быть уже определено или его необходимо вычислить с помощью выражения.

1.5. РЕКОМЕНДАЦИИ ПО РАЗРАБОТКЕ АЛГОРИТМОВ

Чтобы научиться составлять алгоритмы, прежде всего необходимо желание делать это самостоятельно.

При рассмотрении приведенных в пособии примеров для начала внимательно прочитайте условие предложенной задачи и попытайтесь его понять. Затем ответьте на вопрос, как решить задачу математически (известно, что любую задачу можно трактовать математически). А уже затем посмотрите, какое решение предлагается в пособии.

При разработке алгоритмов необходимо руководствоваться следующими **рекомендациями**:

- 1) приступая к разработке алгоритма (написанию программы), четко определите, что является исходными данными и что требуется получить в результате;
- 2) давайте переменным имена, отражающие их назначение;
- 3) для обеспечения универсальности алгоритма, **все известные данные вводите с клавиатуры**, а не задавайте им конкретные значения;
- 4) в алгоритме всегда **предусматривайте «защиту от дурака»**, то есть проверяйте вводимые с клавиатуры данные на допустимость значений. В случае если пользователь ввел недопустимое значение (исходя из условия задачи), необходимо запросить значение заново;
- 5) для контроля вводимых данных сразу же после ввода выводите исходные данные на дисплей;
- 6) в алгоритме **отделяйте ввод данных, вычисления и вывод данных**;
- 7) при записи выражений обращайте внимание на приоритет операций;

8) в блок-схемах при записи в математических формулах действий возведения в степень, квадратного корня и т.п., а также при записи индексов элементов массивов *используйте обычную математическую запись* без привязки к конкретному языку программирования;

9) снабжайте алгоритм (текст программы) содержательными комментариями;

10) после разработки алгоритма выполните его пошагово вручную, чтобы проверить. Для этого еще во время разработки алгоритма готовьте тестовые примеры, содержащие исходные данные и ожидаемые результаты. Отдельно проверьте алгоритм на неверные исходные данные.

1.6. СТРУКТУРЫ АЛГОРИТМОВ И СПОСОБЫ ИХ ЗАПИСИ В ВИДЕ БЛОК-СХЕМЫ

Одним из свойств алгоритма является дискретность – возможность расчленения процесса вычислений, предписанных алгоритмом, на отдельные этапы, возможность выделения участков алгоритма с определенной структурой.

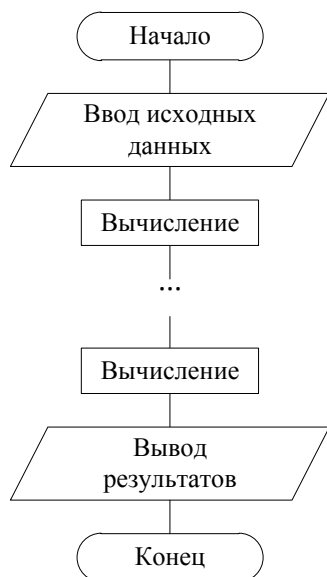


Рис. 1.2. Размещение блоков в линейном алгоритме

Любой алгоритм можно представить как комбинацию трёх элементарных алгоритмических структур, которые называются:

- следование (последовательность двух или более операций);
- ветвление (выбор направления);
- повторение (цикл).

1.6.1. АЛГОРИТМЫ ЛИНЕЙНОЙ СТРУКТУРЫ (СЛЕДОВАНИЕ)

Линейный алгоритм – это алгоритм, в котором все операции выполняются последовательно одна за другой в порядке их записи, их последовательность никогда не нарушается (рис. 1.2).

Пример 2

Зная длины трех сторон треугольника, *вычислить площадь и периметр треугольника*.

Дано (входные данные): **a**, **b**, **c** – длины сторон треугольника.

Найти (выходные данные): **S** – площадь треугольника, **P** – периметр.

Для нахождения площади треугольника можно воспользоваться формулой Герона:

$$S = \sqrt{r \cdot (r - a) \cdot (r - b) \cdot (r - c)},$$

где **r** – полупериметр.

Блок-схема алгоритма вычисления площади и периметра треугольника представлена на рисунке 1.3.

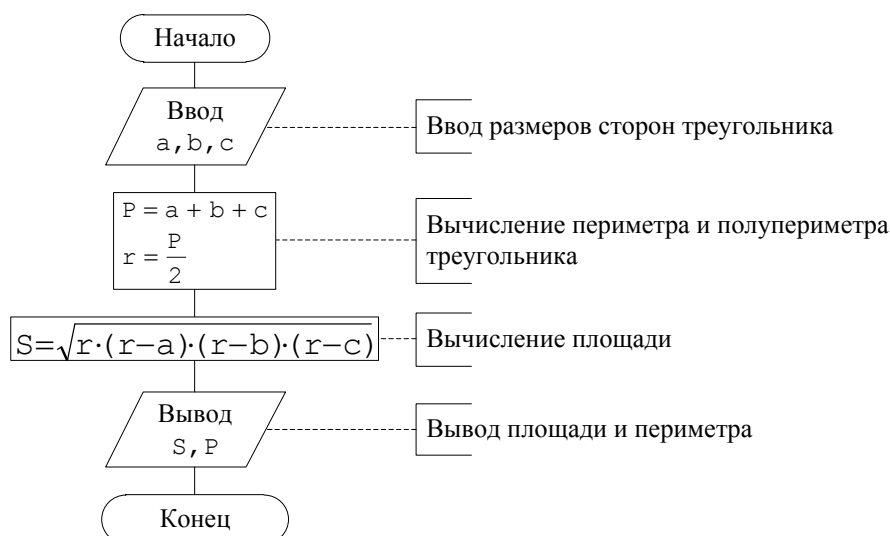


Рисунок 1.3. Вычисление характеристик треугольника

Алгоритм имеет линейную структуру при любых исходных данных – каждое последующее действие следует из предыдущего.

Пример 3

Известны плотность и геометрические размеры цилиндрического слитка, полученного в металлургической лаборатории. **Найти объем, массу и площадь основания слитка.**

Дано (входные данные): **R** – радиус основания цилиндра, **h** – высота цилиндра, **ρ** – плотность материала слитка.

Найти (выходные данные): **m** – массу слитка, **V** – объем, **S** – площадь основания.

Блок-схема вычисления характеристик слитка представлена на рисунке 1.4.

Пример 4

Заданы длины двух катетов в прямоугольном треугольнике. **Найти длину гипотенузы, площадь треугольника и величины его углов.**

Дано (входные данные): **a, b** – длины катетов треугольника.

Найти (выходные данные): **c** – длину гипотенузы, **S** – площадь, **α, β** – углы треугольника.

Геометрическая связь характеристик прямоугольного треугольника представлена на рисунке 1.5. Блок-схема представлена на рисунке 1.6.

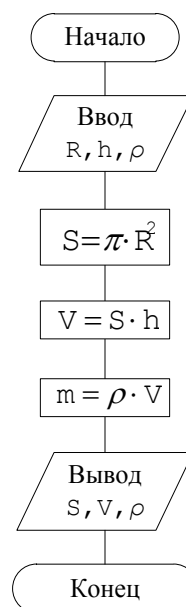


Рис. 1.4. Вычисление характеристик цилиндра

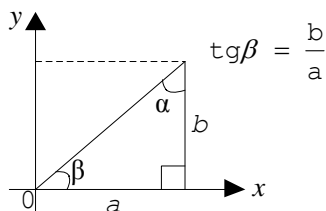


Рис. 1.5. Характеристики прямоугольного треугольника

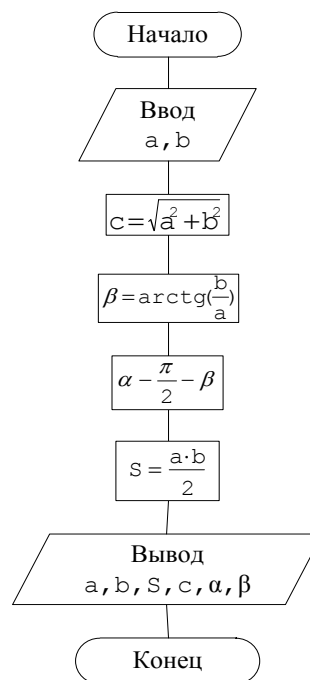


Рис. 1.6. Вычисление характеристик прямоугольного треугольника

1.6.2. АЛГОРИТМЫ РАЗВЕТВЛЕННОЙ СТРУКТУРЫ (ВЕТВЛЕНИЕ)

Алгоритмы разветвленной структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие.

Разветвляющийся алгоритм содержит проверку некоторого логического условия, в зависимости от выполнения или не выполнения которого вычисления могут пойти по разным направлениям.

Условие – вопрос, на который обычно имеется два варианта ответа: ДА (ИСТИНА) или НЕТ (ЛОЖЬ). В блок-схемах разветвленные алгоритмы изображаются так, как показано на рисунках 1.7, 1.8.

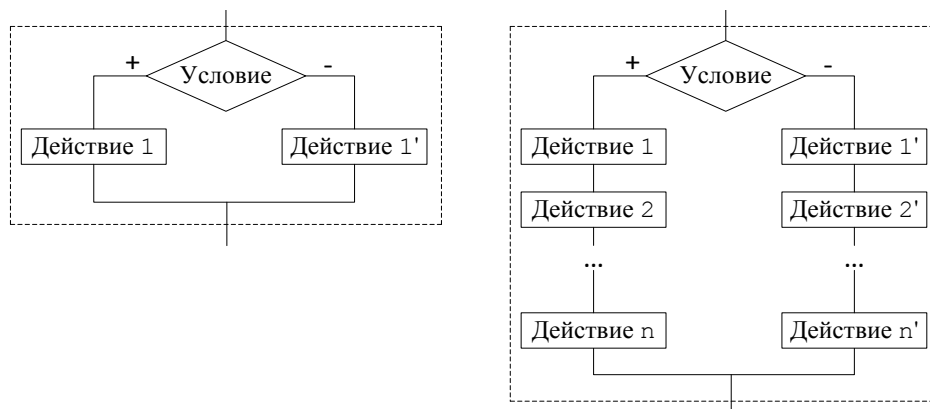


Рис. 1.7. Полная форма разветвляющейся структуры

Условие – вопрос, на который имеется более двух ответов – несколько альтернативных решений, в блок-схемах изображается так, как показано на рисунке 1.9.

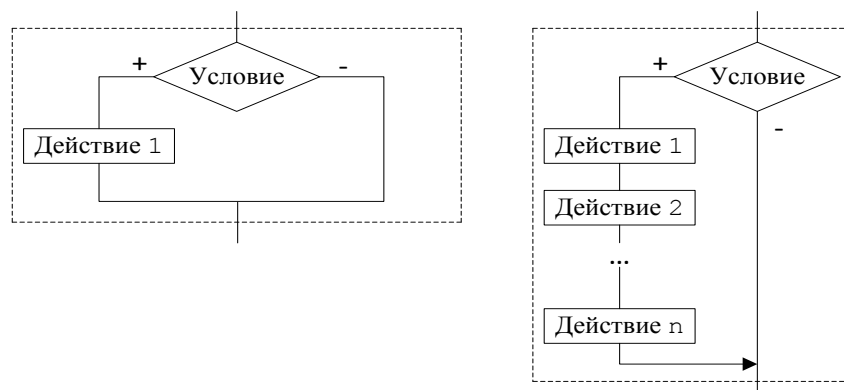


Рис. 1.8. Неполная форма разветвляющейся структуры

С целью получения структурированного алгоритма при использовании в алгоритме любой структуры, в том числе и ветвления, *следите за тем, чтобы любой блок имел только один вход и только один выход. У блока ветвления две ветви +/ДА и –/НЕТ в конечном счете, должны быть соединены в один общий выход* (обратите внимание на такое соединение внизу на рисунках 1.7, 1.8 и 1.9).

На рисунках 1.7 – 1.9 *пунктирными линиями* дополнительно выделены управляющие конструкции. Эти *линии подчеркивают*, что каждая из данных конструкций имеет *один вход и один выход*, поэтому их можно рассматривать как обобщенные функциональные блоки («черные ящики»).

Внимание!!! При включении в алгоритм вычисления значения математического выражения, следует проанализировать, для всех ли значений переменных это выражение можно вычислить.

В алгоритме необходимо предусмотреть предварительную проверку переменных на значения, для которых выражение не может быть определено.

Если, например, требуется вычислить корень четной степени, то нужно перед вычислением проверить подкоренное выражение – оно не должно принимать отрицательные значения, а в случае с дробью – проверить знаменатель на равенство его **НУЛЮ**.

В блок-схеме такие проверки реализуются с помощью условного блока. Отсутствие таких проверок в программе может привести к критическим ошибкам. Учитывая это, алгоритм нахождения корней квадратного уравнения (см. рис. 1.1) необходимо дополнить проверкой коэффициента **а** на равенство **НУЛЮ** (рис. 1.10 и 1.11).

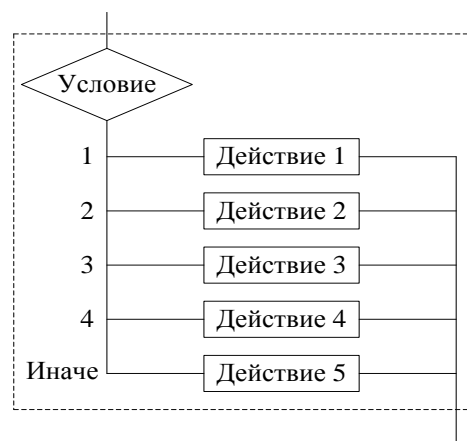


Рис. 1.9. Множественный выбор

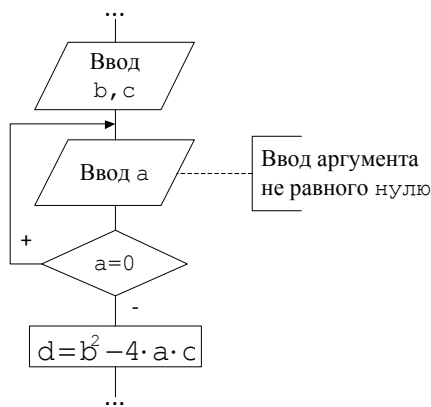


Рис. 1.10. Повторение ввода переменной **a**

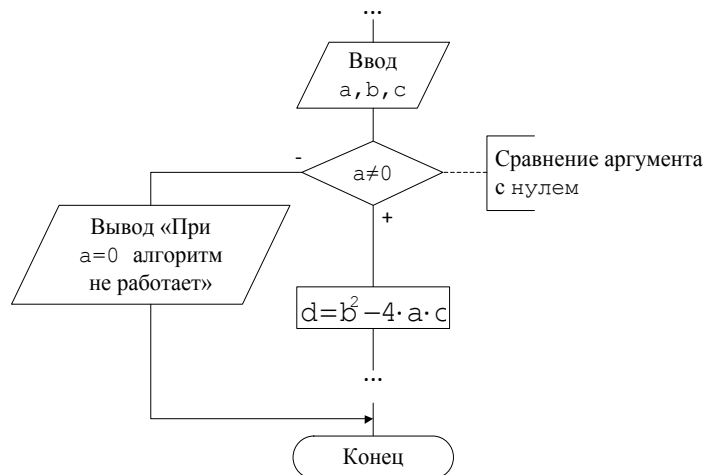


Рис. 1.11. Проверка входных данных

Рассмотрим примеры построения алгоритмов разветвленной структуры.

Пример 5

Найти наименьшее из трех чисел.

Дано (входные данные): числа **a, b, c**.

Найти (выходные данные): **m** – число, равное наименьшему из чисел **a, b, c**.

Варианты решения задачи приведены на рисунке 1.12.

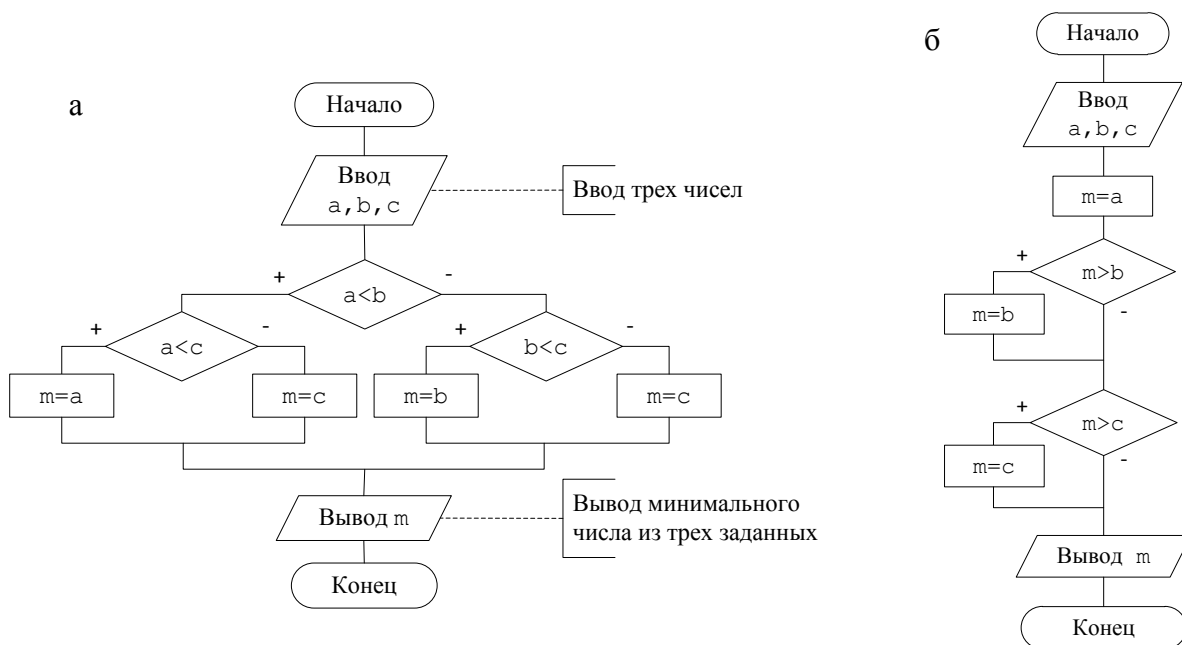


Рис. 1.12. Нахождение наименьшего из трех чисел:
а - 1-й вариант решения; б - 2-й вариант решения

Обратите внимание, у ветвлений четко соединены альтернативные пути в один до перехода к следующему блоку, чтобы каждый **блок имел только один вход и только один выход**. Это сделано с целью получения структурированного алгоритма.

Пример 6

Функция $y=f(x)$ задана графически (рис. 1.13).

Определить значение функции для произвольно заданного вещественного x .

Дано (входные данные): x .

Найти (выходные данные): y .

В данной задаче модель задана в виде графика. Запишем ее в виде формул.

Из графика (рис. 1.13) следует, что вся числовая ось разбита на три части:

$$x < -1, -1 \leq x \leq 1, x > 1.$$

На полуинтервале $x < -1$ задана прямая, параллельная оси x . Её уравнение $y=1$.

На интервале $-1 \leq x \leq 1$ уравнение прямой: $y=|x|$.

На полуинтервале $x > 1$ функция не определена.

Математическую модель задачи можно записать так:

$$y = f(x) = \begin{cases} 1, & \text{если } x < -1; \\ |x|, & \text{если } -1 \leq x \leq 1; \\ \text{Функция не определена,} & \text{если } x > 1. \end{cases}$$

«Неудачными» вариантами решения данной задачи являются алгоритмы, приведенные на рисунке 1.14. Недостатком этих алгоритмов является то, что в них не учтены указанные ранее рекомендации: **«в алгоритме отделяй те ввод данных, вычисления и вывод данных»**.

Если сравнивать между собой алгоритмы двух вариантов, то в варианте (см. рис. 14, а) составлен менее «неудачный» алгоритм, так как он является структурированным. А в варианте (см. рис. 14, б) составлен не структурированный алгоритм, так как ветвления в нем не удовлетворяют требованию, что каждый **блок должен иметь только один вход и только один выход**, поэтому нет возможности выделить конструкции ветвления пунктирными линиями.

С целью исправления допущенных ошибок, введем в алгоритм так называемый «флаг» («признак») – переменную **F**.

В качестве «флага» в алгоритмах часто используют переменные, которым программист сам присваивает определенные значения. Эти значения придумываются (оговариваются) заранее и вводятся в соответствующих местах программ по мере необходимости. «Флаг» часто используют с целью получения структурированного алгоритма, его использование позволяет обходиться без оператора **ГОТО**.

В данной задаче договариваемся, что «флаг» **F** будет принимать значение **1**, если функция будет определена, и **F** будет равен **0**, если функция не определена (рис. 1.15). Обратите внимание на выделение в алгоритме пунктирными линиями конструкций ветвления, каждая имеет только один вход и только один выход.

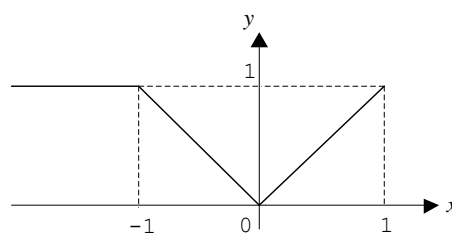
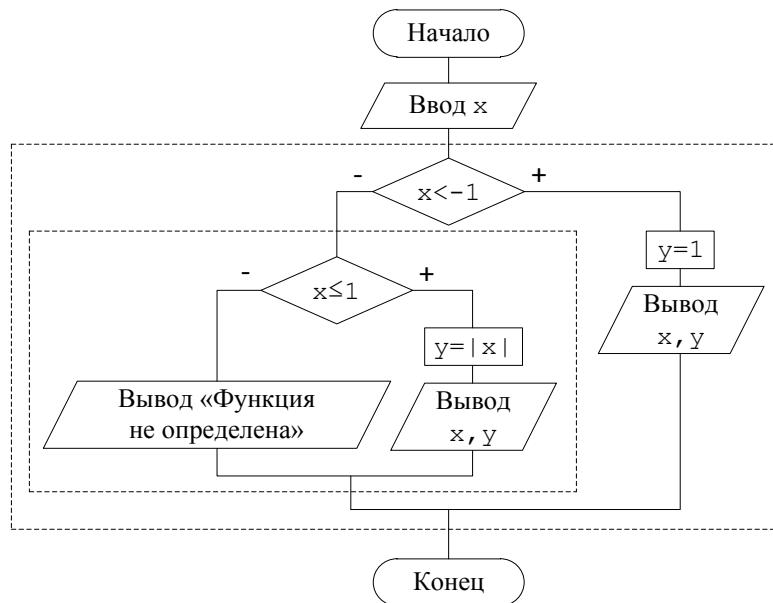


Рис. 1.13. Графическое представление функции (см. пример 6)

а



б

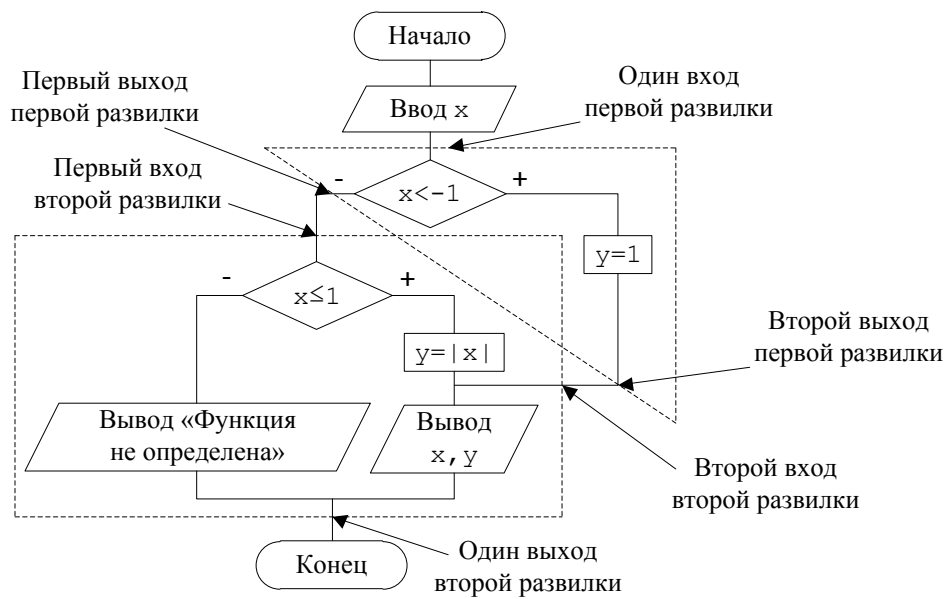


Рис. 1.14. «Неудачные» алгоритмы определения значения функции, заданной графически:

а - не отделены вычисления и вывод результатов;

б - не структурированный алгоритм и не отделены вычисления и вывод результатов

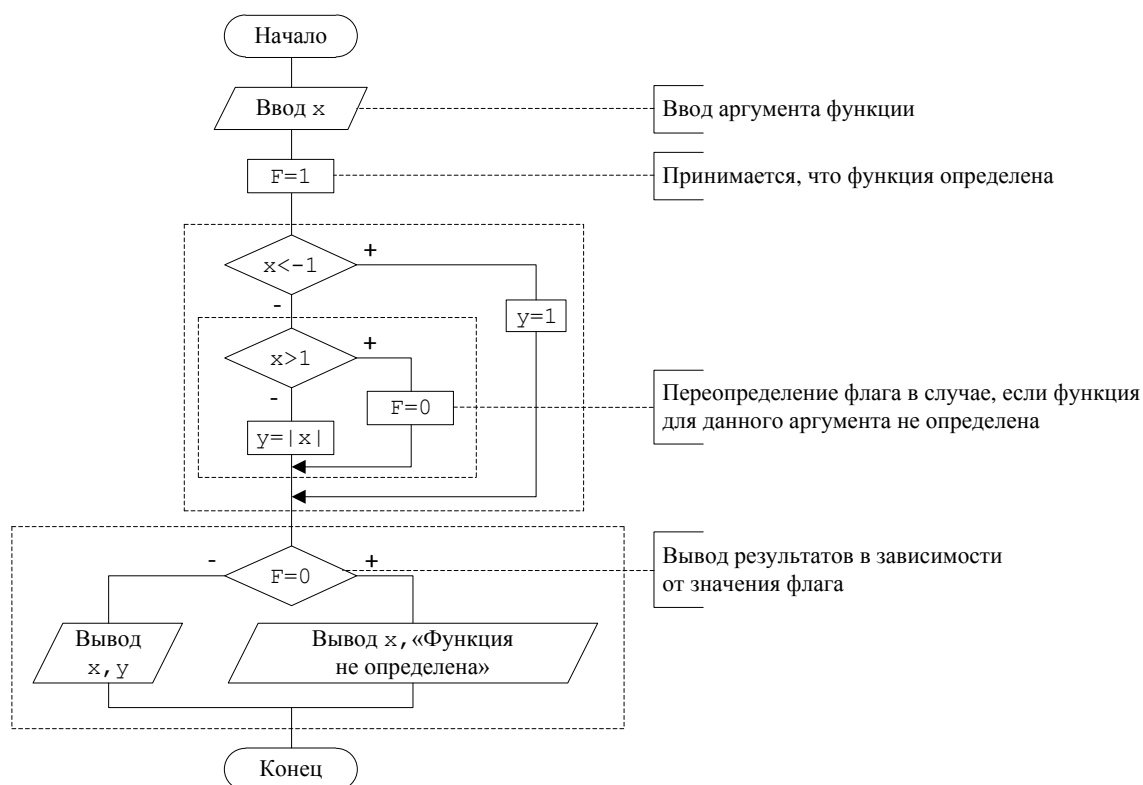


Рис. 1.15. «Удачный» алгоритм определения значения функции, заданной графически

Пример 7

Функция $y=f(x)$ задана графически (рис. 1.16).

Определить значение функции для произвольно заданного вещественного x .

Дано (входные данные): x .

Найти (выходные данные): y .

В данной задаче модель задана в виде графика. Запишем ее в виде формул.

Из графика следует, что вся числовая ось разбита на три части:

$$x < -2, -2 \leq x \leq 2, x > 2.$$

На полуинтервале $x < -2$ функция не определена.

На интервале $-2 \leq x \leq 2$ задана окружность. Уравнение окружности:

$$y = \sqrt{4 - x^2}.$$

На полуинтервале $x > 2$ задана прямая линия, ее уравнение:

$$y = x - 2.$$

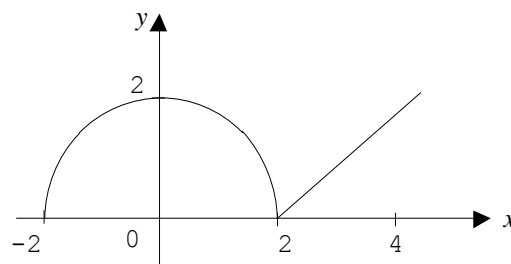


Рис. 1.16. Графическое представление функции (см. пример 7)

Получили его так: прямая линия проходит через две точки с координатами $(2, 0)$ и $(4, 2)$. Уравнение прямой, проходящей через эти точки, имеет вид:

$$\frac{x - x_1}{x_1 - x_2} = \frac{y - y_1}{y_1 - y_2}.$$

тогда

$$\frac{x - 2}{2 - 4} = \frac{y - 0}{0 - 2}.$$

После преобразований последнего выражения получим:

$$y = x - 2.$$

Математическую модель задачи можно записать так:

$$y = f(x) = \begin{cases} \text{Функция не определена, если } x < -2; \\ \sqrt{4 - x^2}, & \text{если } -2 \leq x \leq 2; \\ x - 2, & \text{если } x > 2. \end{cases}$$

Алгоритм определения значения функции в виде блок-схемы аналогичен алгоритму решения предыдущего примера (см. рис. 1.15).

Пример 8

Определить, попадает ли точка с произвольными координатами (x, y) в заштрихованную область (рис. 1.17).

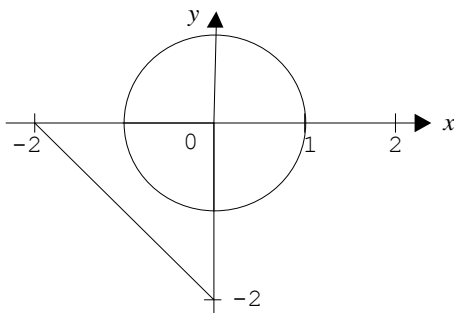


Рис. 1.17. Заданная область (см. пример 8)

Дано (входные данные): x, y – координаты, **область**, ограниченная функциями.

Найти (выходные данные): **сообщение о попадании точки в область** – «Да» или «Нет».

Для решения данной задачи надо, учитывая заданную область, записать условие попадания точки в эту область, то есть построить математическую модель.

Запишем условия попадания точки в область в виде формул.

Область можно описать как круг, пересекающийся с треугольником. Точка может попасть либо в круг, либо в треугольник, либо в их общую часть.

Условие попадания точки в круг имеет вид:

$$x^2 + y^2 \leq 1.$$

Условия попадания точки в треугольник:

$$x \leq 0, y \leq 0, y \geq -x - 2.$$

Уравнение $y = -x - 2$ наклонной прямой, ограничивающей треугольную область, получили следующим образом. Прямая линия проходит через две точки с координатами $(-2, 0)$ и $(0, -2)$. Уравнение прямой, проходящей через эти точки имеет вид:

$$\frac{x - x_1}{x_1 - x_2} = \frac{y - y_1}{y_1 - y_2},$$

тогда

$$\frac{x - (-2)}{-2 - 0} = \frac{y - 0}{0 - (-2)}.$$

После преобразований последнего выражения получим:

$$y = -x - 2.$$

Координаты точек, лежащих выше этой наклонной прямой, удовлетворяют условию:

$$y \geq -x - 2.$$

Точка с заданными координатами (x, y) попадет в заштрихованную область, если эти координаты будут удовлетворять или первому условию, или второму.

Схема алгоритма определения попадания точки в заданную область приведена на рисунке 1.18.

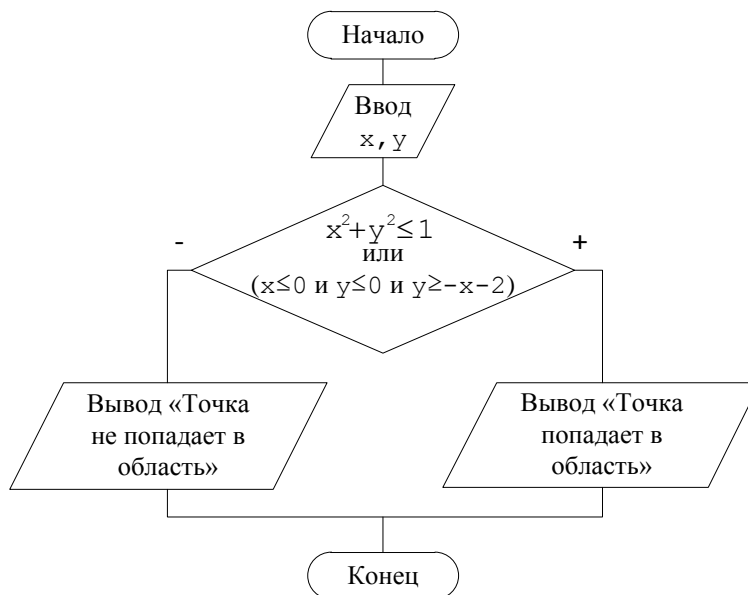


Рис. 1.18. Определение попадания точки в заданную область

1.6.3. АЛГОРИТМЫ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ (ПОВТОРЕНИЕ)

Циклический алгоритм содержит повторяющиеся участки вычислений. Повторяющийся участок называют **циклом**.

Циклы различаются *по способам организации и по содержанию* (рис. 1.19).

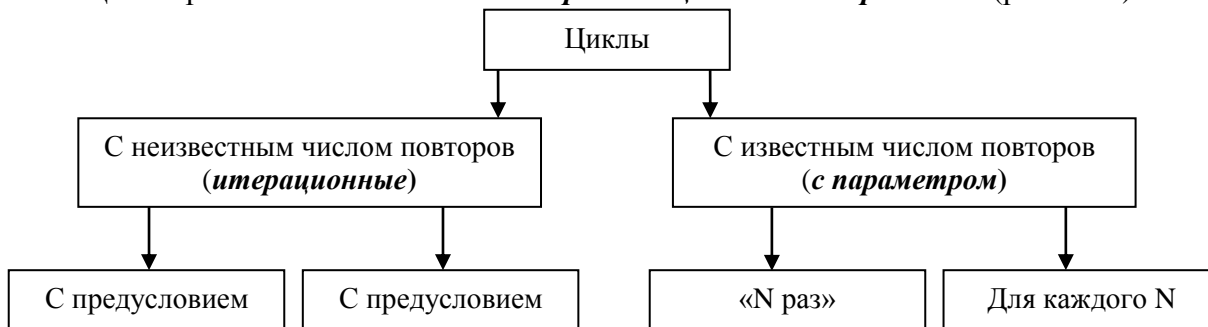


Рис. 1.19. Классификация циклов

По содержанию циклы делятся на циклы:

- с неизвестным числом повторов (*итерационные*);
- с известным числом повторов (*с параметром*).

По способу организации и исполнения проверки условия окончания цикла различают **две разновидности циклических структур**:

- **циклы с предусловием;**
- **циклы с постусловием.**

Соответственно используются два варианта конструкции повторения: для реализации цикла с «предусловием» и с «постусловием» (рис. 1.20).

Телом цикла называется часть цикла, не включающая проверку условия окончания цикла.

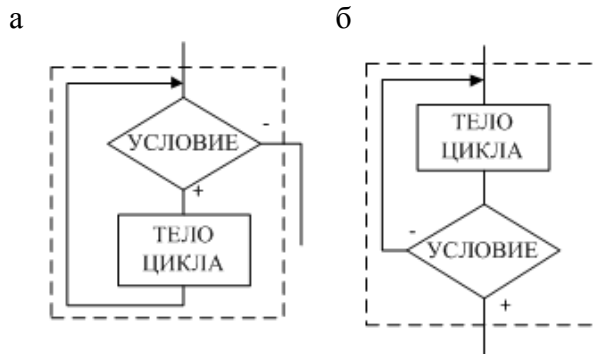


Рис. 1.20. Алгоритм циклической структуры:
а – с предусловием; б – с постусловием

В цикле с «предусловием» условие продолжения циклического процесса ставится в начале цикла, в цикле с «постусловием» – в конце. В обоих случаях цикл повторяется до тех пор, пока значение условия остается ИСТИНА.

В современных алгоритмических языках есть операторы, позволяющие реализовать эти конструкции в случаях, когда ДА и НЕТ меняются местами.

В цикле с «предусловием» в частном случае может оказаться, что действие не выполнялось ни разу.

В цикле с «постусловием» тело цикла будет реализовано хотя бы один раз. После этого происходит проверка условия.

Условие цикла необходимо подобрать так, чтобы действия, выполняемые в цикле, привели к нарушению его истинности, иначе произойдет заикливание.

Заикливание – бесконечное повторение выполняемых действий.

При написании условных циклических алгоритмов следует помнить следующее:

- во-первых, чтобы цикл имел шанс когда-нибудь закончиться, **содержимое его тела должно обязательно влиять на условие** цикла;
- во-вторых, **условие должно состоять из корректных выражений и значений, определенных еще до первого выполнения тела цикла.**

На рисунке 1.20 **пунктирными линиями** дополнительно выделены управляющие конструкции. Эти **линии подчеркивают, что каждая конструкция имеет один вход и один выход**, поэтому их можно рассматривать как обобщенные функциональные блоки («черные ящики»). Внутри этих конструкций, в свою очередь, могут находиться функциональные блоки, которые могут быть, в частности, конструкциями такого же типа, то есть возможны «вложенные» развилки и повторения или их комбинации. Но какова бы не была глубина вложенности, важно, что любая конструкция в конечном итоге имеет один вход и один выход с целью получения структурированного алгоритма.

При конструировании схем и программ из таких блоков выход из одного блока подключается к входу другого блока, в результате схема и программа имеют последовательную структуру.

Таким образом, **по содержанию циклы** делятся еще на циклы:

- **простые** – не содержат внутри себя других циклов;
- **сложные (иначе – вложенные)** – содержат внутри себя, по крайней мере, хотя бы один цикл.

На рисунке 1.20 представлены **циклы с неизвестным числом повторов**, они применяются, когда заранее неизвестно, сколько раз повторится тело цикла (вообще говоря, неизвестно, закончится ли цикл вообще).

Цикл, количество повторений которого известно заранее или его можно определить по исходным данным, называют **циклом с известным числом повторов**.

Цикл с известным числом повторов, или цикл с параметром, или цикл со счетчиком, или арифметический цикл – это цикл, который организован по некоторой переменной, для которой известны начальное значение, конечное значение и шаг изменения (рис. 1.21 и 1.22).

Если начальное значение и шаг изменения равны единице, то параметр цикла одновременно является **счетчиком цикла**. Тогда его конечное значение определяет количество повторений цикла. Для цикла с параметром характерно то, что всегда можно определить **число его повторений**.

Эту переменную называют **параметром цикла**.

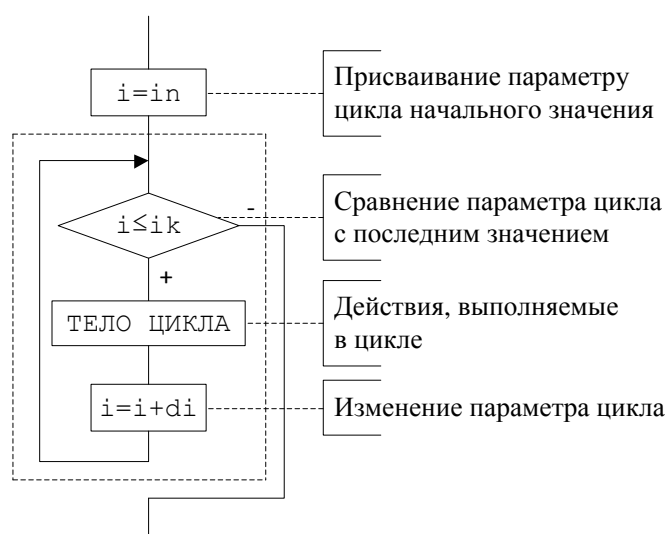


Рис. 1.21. Условный циклический алгоритм с известным числом повторений

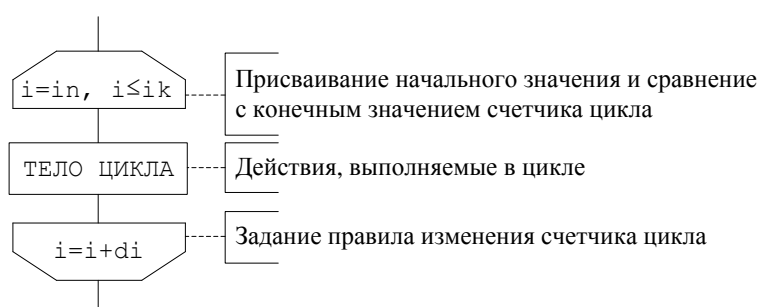


Рис. 1.22. Алгоритм циклической структуры без условия

В блоке модификации указывается закон изменения переменной параметра: **in** – начальное значение параметра; **ik** – последнее значение параметра; **di** – шаг.

Рассмотрим использование алгоритмов циклической структуры на конкретных примерах.

Пример 9

Найти наибольший общий делитель (НОД) двух натуральных чисел **A** и **B**.

Дано (входные данные): **A > 0** и **B > 0**.

Найти (выходные данные): **число**, равное наибольшему общему делителю.

Для решения поставленной задачи воспользуемся алгоритмом Евклида: будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба значения не станут равными, как показано в таблице 1.2.

Таблица 1.2

Вычисление наибольшего общего делителя

Шаг алгоритма	Проверка условия выхода из цикла A ≠ B	A	B
Исходные данные		A=25	B=15
1-й шаг цикла	25 ≠ 15 ДА	A=A-B=10	
2-й шаг цикла	10 ≠ 15 ДА		B=B-A=5
3-й шаг цикла	10 ≠ 5 ДА	A=A-B=5	
Выход из цикла	5 ≠ 5 НЕТ		
Результат		НОД чисел A и B равен 5	

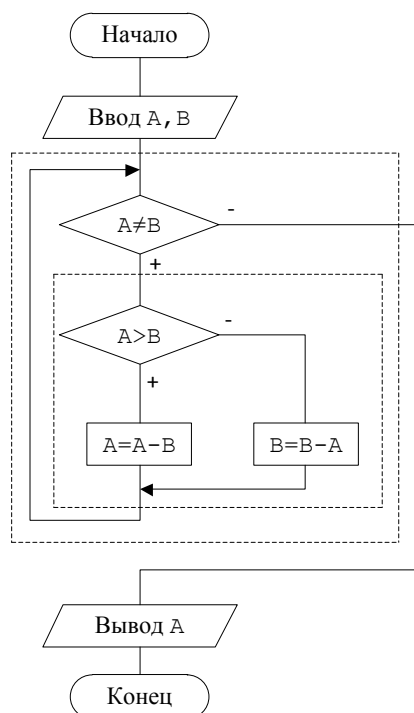


Рис. 1.23. Поиск наибольшего общего делителя двух чисел

В блок-схеме (рис. 1.23) для решения поставленной задачи используется цикл с предусловием. Тело цикла повторяется пока **A** не равно **B**.

Обратите внимание на выделение в алгоритме пунктирными линиями конструкций ветвления и цикла, каждая имеет только один вход и только один выход.

В этом алгоритме не хватает «защиты от дурака», то есть проверки вводимых чисел:

- если **A ≤ 0** или **B ≤ 0**, то такое число надо ввести заново;
- в остальных случаях можно считать по данному алгоритму.

Пример 10

Вводится последовательность чисел, **0** – конец последовательности. **Определить, содержит ли последовательность хотя бы два равных соседних числа.**

Дано (входные данные): **x0** – текущий член последовательности, **x1** – следующий член последовательности.

Найти (выходные данные): *сообщение о наличии* в последовательности чисел *двух равных соседних элементов*.

Перед разработкой алгоритма полезно составить тестовый пример, чтобы более наглядно представить себе алгоритм.

Составим несколько тестовых примеров.

Последовательности чисел:

Есть два равных соседних числа	5	7	8	25	0
Нет двух равных соседних чисел	5	8	8	25	0

Введем вспомогательную переменную **F** («флаг») – логическую переменную, которая будет сохранять значение ЛОЖЬ, если в последовательности нет равных рядом стоящих элементов, и будет получать значение ИСТИНА, если найдутся в последовательности два равных соседних числа.

Блок-схема решения задачи приведена на рисунке 1.24. Применение здесь цикла с постусловием обосновано тем, что необходимо вначале сравнить два элемента последовательности, а затем принять решение об окончании цикла.

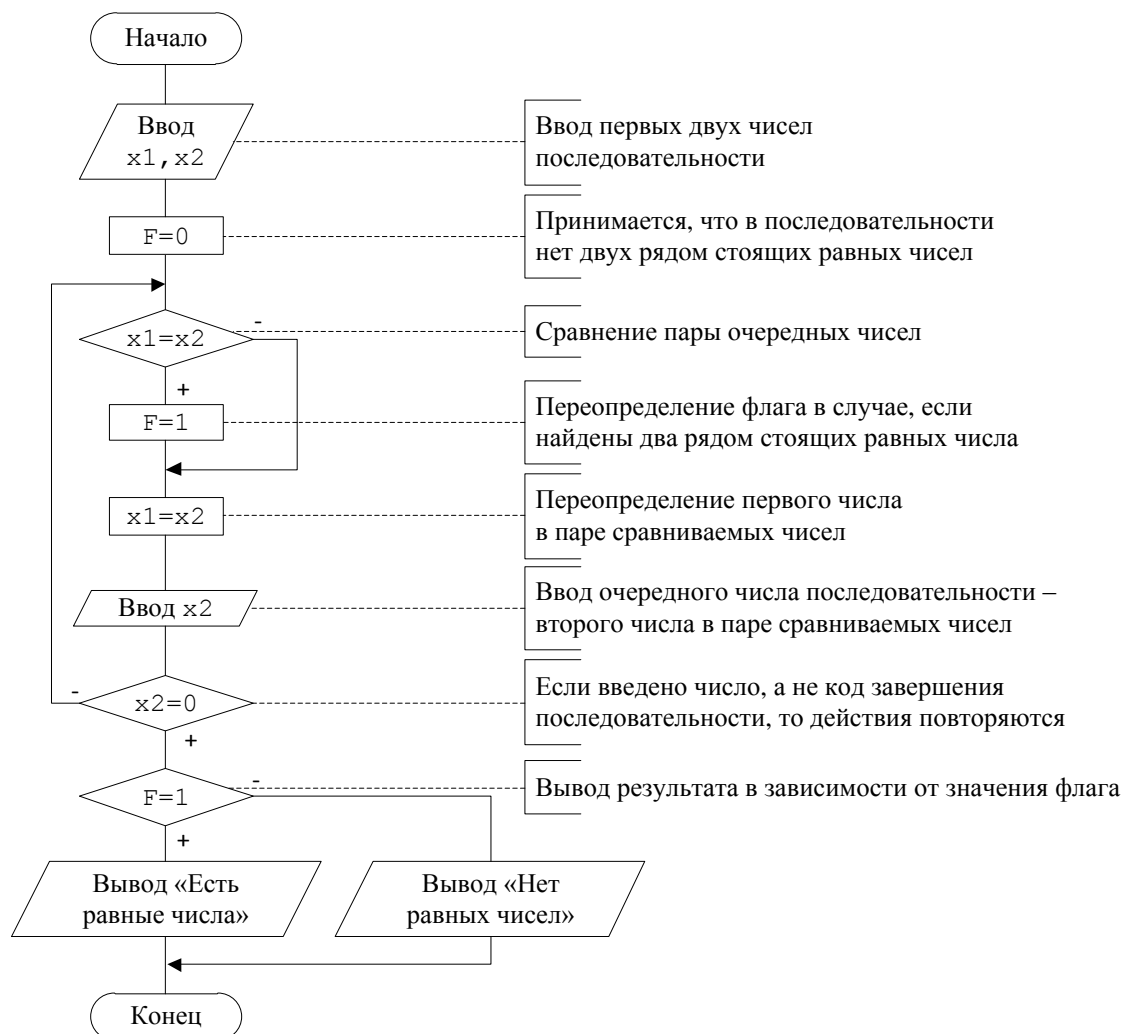


Рис. 1.24. Поиск равных соседних элементов последовательности

Выполним «ручную отладку» алгоритма для последовательности из четырех чисел **5, 7, 8, 25** и завершающего нуля, то есть проверим то, как будет работать алгоритм для последовательности, в которой нет рядом равных чисел (табл. 1.3).

Таблица 1.3

Выявление наличия в последовательности чисел **5, 7, 8, 25, 0**
двух равных соседних элементов

Шаг алгоритма	Проверка равенства двух соседних элементов $x_0 = x_1$	x_0	x_1	F	Проверка условия выхода из цикла $x_1 = 0$
До цикла		5	7	ЛОЖЬ	
1-й шаг цикла	$5 = 7$ НЕТ	$x_0=x_1=7$	8		$8 = 0$ НЕТ
2-й шаг цикла	$7 = 8$ НЕТ	$x_0=x_1=8$	25		$25 = 0$ НЕТ
3-й шаг цикла	$8 = 25$ НЕТ	$x_0=x_1=25$	0		$0 = 0$ ДА
Результат	Нет равных чисел				

Выполним «ручную отладку» алгоритма для последовательности из четырех чисел **5, 8, 8, 25** и завершающего нуля, то есть проверим то, как будет работать алгоритм для последовательности, в которой есть рядом равные числа (табл. 1.4).

Таблица 1.4

Выявление наличия в последовательности чисел **5, 8, 8, 25, 0**
двух равных соседних элементов

Шаг алгоритма	Проверка равенства двух соседних элементов $x_0 = x_1$	x_0	x_1	F	Проверка условия выхода из цикла $x_1 = 0$
До цикла		5	8	ЛОЖЬ	
1-й шаг цикла	$5 = 8$ НЕТ	$x_0=x_1=8$	8		$8 = 0$ НЕТ
2-й шаг цикла	$8 = 8$ ДА	$x_0=x_1=8$	25	ИСТИНА	$25 = 0$ НЕТ
3-й шаг цикла	$8 = 25$ НЕТ	$x_0=x_1=25$	0		$0 = 0$ ДА
Результат	Есть равные числа				

В приведенных примерах 9 и 10 условия задач такие, что неизвестно, сколько раз повторится тело цикла, поэтому в алгоритмах был использован **цикл с неизвестным числом повторений**.

Рассмотрим несколько примеров с использованием **циклов с известным числом повторений**.

Пример 11

Составить таблицу значений функции $y=e^{\sin(x)} \cos(x)$ на отрезке $[0; \pi]$ с шагом $0,1$.

Дано (входные данные): **начальное** значение аргумента $X - 0$, **конечное** значение аргумента $- \pi$, **шаг** изменения аргумента $- 0,1$.

Найти (выходные данные): множество значений аргумента X и соответствующее им множество значений функции Y .

В условии задачи, с одной стороны, количество повторений цикла явно не задано, поэтому решить ее можно, используя цикл с предусловием (рис. 1.25, а).

С другой стороны, известно, как изменяется параметр цикла X и каковы его начальное и конечное значения, следовательно, предварительно определив количество повторений тела цикла (n), можно воспользоваться безусловным циклическим оператором (рис. 1.25, б).

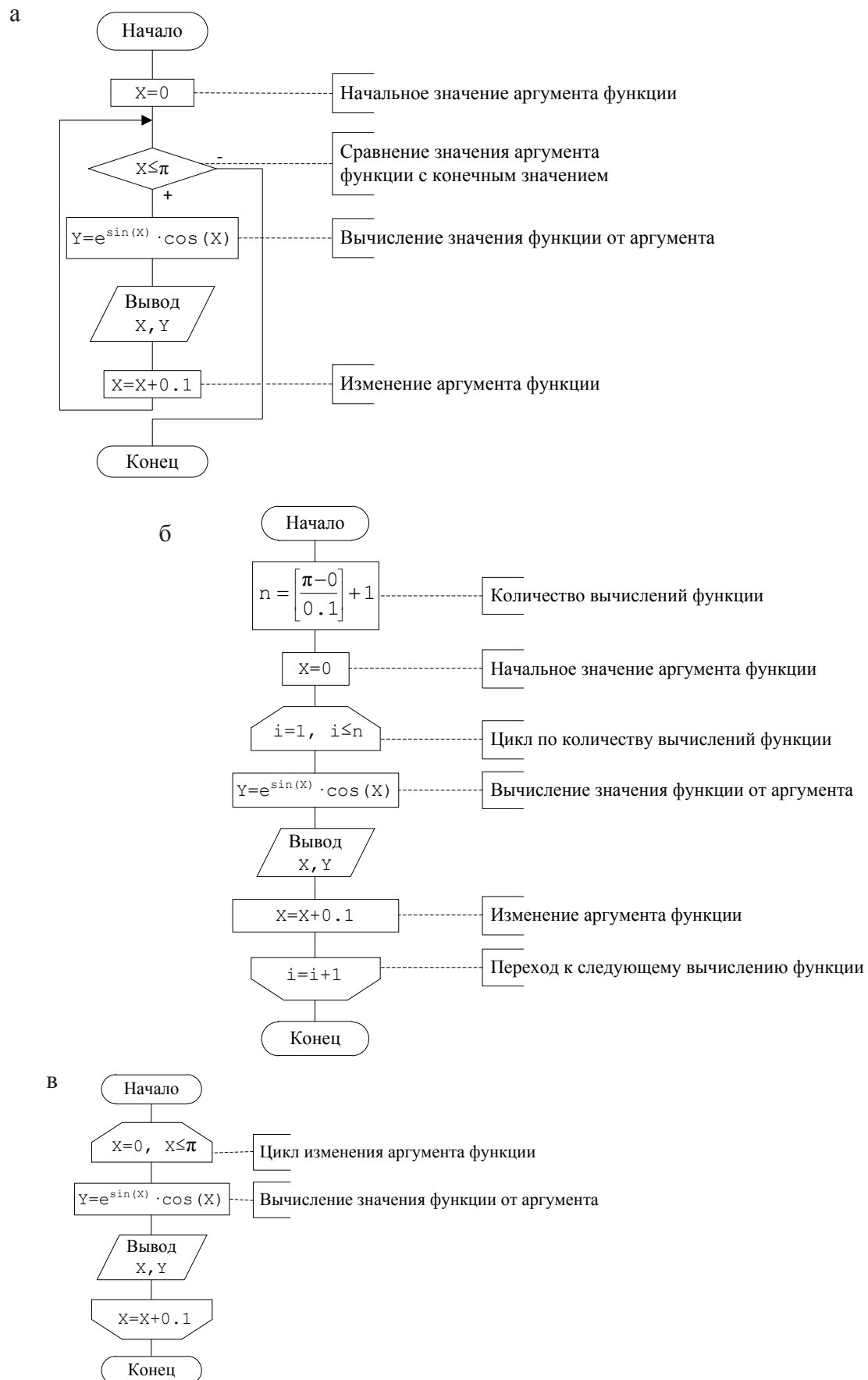


Рис. 1.25. Создание таблицы значений функции $y = e^{\sin(x)} \cos(x)$:
а – 1-й способ; б – 2-й способ; в – 3-й способ

Итак, если параметр цикла **X** принимает значения в диапазоне от **xn** до **xk**, изменяясь с шагом **dx**, то количество повторений тела цикла можно определить по формуле:

$$n = \left\lfloor \frac{xk - xn}{dx} \right\rfloor + 1,$$

округлив результат деления до целого числа в меньшую сторону (операция $\lfloor \rfloor$ – округление до наименьшего целого).

Цикл можно организовать еще проще – по переменной **X** – аргументу функции (рис. 1.25, в).

Пример 12

Вычислить факториал числа N ($N! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot N$).

Дано (входные данные): **N** – натуральное число, факториал которого необходимо вычислить.

Найти (выходные данные): **P** (целое число) – значение факториала числа **N**, произведение чисел от 1 до **N**.

Математическая модель задачи:

Вычислить $P=N!$, где $N \geq 0$ – целое.

Воспользуемся формулой вычисления факториала:

$$P = N! = \begin{cases} 1 & , \text{ если } N = 0; \\ 1 \cdot 2 \cdot 3 \cdot \dots \cdot N & , \text{ если } N > 0. \end{cases}$$

Итак, после ввода числа **N** необходимо поочередно перебирать (брать) целые числа от **2** до **N** и умножать на них переменную **P**, в которой будет храниться произведение этих чисел, то есть факториал.

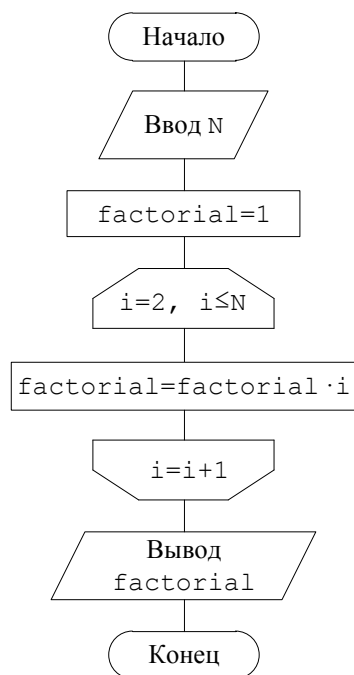


Рис. 1.26. Вычисление факториала

Переменной **P**, предназначенной для хранения значения произведения последовательности чисел, присваивается начальное значение, равное единице, не влияющее на конечный результат.

Затем организуется цикл, параметром которого выступает целочисленная переменная **i**, принимающая значения от **2** до **N** с шагом **1**, параметр цикла.

Блок-схема вычисления факториала приведена на рисунке 1.26.

Если значение параметра цикла меньше или равно **N**, то выполняется оператор тела цикла, в котором из участка памяти с именем **P** считывается предыдущее значение произведения, умножается на текущее значение параметра цикла **i**, а результат снова помещается в участок памяти с именем **P**. Когда параметр **i** становится больше **N**, цикл заканчивается, и на печать выводится значение переменной **P**, которая была вычислена в теле цикла.

В этом алгоритме хорошо бы добавить «защи-

ту от дурака – проверить, если $N < 0$, то его надо ввести заново, в остальных случаях можно считать по данному алгоритму.

Если будет введено $N=0$, то данный алгоритм выдаст верный ответ – **1**, хотя никаких дополнительных условий не введено. Верный ответ при $N=0$ получится, потому что до цикла переменной **factorial** задано значение **1**, а при входе в цикл в блоке начала цикла переменная **i** примет значение **2**, что больше, чем $N=0$, и первая же проверка даст **ЛОЖЬ**. Таким образом, тело цикла не будет выполнено ни одного раза и переменная **factorial** сохранит свое первоначальное значение **1**.

Выполним «ручную отладку» алгоритма, то есть проверим то, как будет работать алгоритм при каждом повторении цикла не за ПЭВМ, а на листе бумаги.

Такую отладку настоятельно рекомендуется выполнять для каждого вновь разрабатываемого алгоритма, особенно начинающим программистам. Не обязательно расписывать все повторения цикла, да это и невозможно, если он повторяется много раз. Обычно достаточно расписать несколько первых повторений цикла, а в случае цикла с параметром – несколько первых и несколько последних.

В таблице 1.5 отображен протокол выполнения алгоритма при вычислении факториала числа $n=5$.

Таблица 1.5

Вычисление факториала числа

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	factorial
До цикла			factorial=1
1-й шаг цикла	$i=2$	$2 \leq 5$ ДА	factorial=factorial·i= 1·2= 2
2-й шаг цикла	$i=3$	$3 \leq 5$ ДА	factorial=factorial·i= 2·3= 6
3-й шаг цикла	$i=4$	$4 \leq 5$ ДА	factorial=factorial·i= 6·4= 24
4-й шаг цикла	$i=5$	$5 \leq 5$ ДА	factorial=factorial·i=24·5=120
Выход из цикла	$i=6$	$6 \leq 5$ НЕТ	
Результат			factorial=120

Пример 13

Вычислить a^n ($n > 0$).

Дано (входные данные): **a** – вещественное число, которое необходимо возвести в степень; **n** – целая положительная степень.

Найти (выходные данные): **P** (вещественное число) – результат возведения вещественного числа **a** в целую положительную степень **n**.

Известно, для того, чтобы получить целую степень **n** числа **a**, нужно умножить его само на себя **n** раз. Поэтому организуем цикл по целочисленной переменной **i** (параметру цикла), которая будет принимать значения от **1** до **n** с шагом **1**.

Цикл выполняется **n** раз.

Результат умножения будет храниться в участке памяти с именем **P**. При выполнении очередного шага цикла из этого участка памяти предыдущее значение будет считываться, умножаться на основание степени **a** и снова записываться в участок памяти **P**.

Для выполнения первого оператора накапливания произведения из участка памяти необходимо взять такое число, которое не влияло бы на результат умножения. То

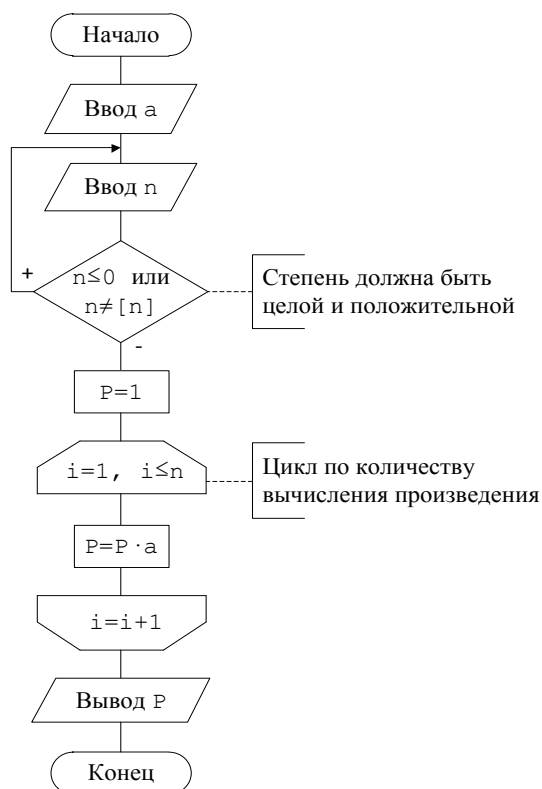


Рис. 1.27. Возведение вещественного числа
в целую степень

есть перед началом цикла переменной **P**, предназначенной для накапливания произведения, необходимо присвоить значение *единица*.

Блок-схема возведения числа в степень приведена на рисунке 1.27. В этот алгоритм добавлена «защита от дурака» – проверка степени **n**, чтобы она была больше 0 и целая. Для проверки целостности числа используется операция **[]** – округление числа до наименьшего целого.

В таблице 1.6 отображен протокол выполнения алгоритма при возведении числа два в пятую степень: **a=2, n=5**.

Таблица 1.6

Возведение числа в степень

Шаг алгоритма	i	Проверка условия выхода из цикла i ≤ n	P
До цикла			P=1
1-й шаг цикла	i=1	1 ≤ 5 ДА	P=P·a= 1·2= 2
2-й шаг цикла	i=2	2 ≤ 5 ДА	P=P·a= 2·2= 4
3-й шаг цикла	i=3	3 ≤ 5 ДА	P=P·a= 4·2= 8
4-й шаг цикла	i=4	4 ≤ 5 ДА	P=P·a= 8·2=16
5-й шаг цикла	i=5	5 ≤ 5 ДА	P=P·a=16·2=32
Выход из цикла	i=6	6 ≤ 5 НЕТ	
Результат			P=32

Пример 14

Вычислить сумму натуральных четных чисел, не превышающих N.

Дано (входные данные): **N** – целое число больше **НУЛЯ**.

Найти (выходные данные): **S** – сумму четных чисел.

При сложении нескольких чисел необходимо накапливать результат в определенном участке памяти **S**, каждый раз считывая из этого участка предыдущее значение суммы и прибавляя к нему следующее слагаемое.

Для выполнения первого оператора накапливания суммы из участка памяти необходимо взять такое число, которое не влияло бы на результат сложения. То есть перед

началом цикла переменной **S**, предназначенной для накопления суммы, необходимо присвоить значение **НУЛЬ**.

Так как нужно складывать только четные числа, организуем цикл по целочисленной переменной **i**, принимающей заведомо четные значения от **2** до **N** с шагом **2**.

Блок-схема вычисления суммы натуральных четных чисел представлена на рисунке 1.28. В этом алгоритме не хватает «защиты от дурака». Кроме того, при заранее известном законе изменения параметра цикла (переменная **i**), лучше использовать алгоритм циклической структуры без условия, то есть с использованием символов границ

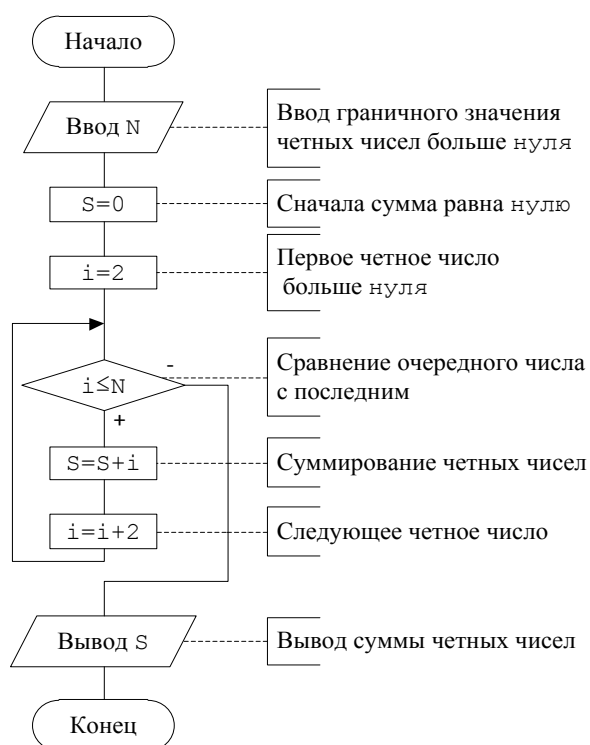


Рис. 1.28. Вычисление суммы четных натуральных чисел

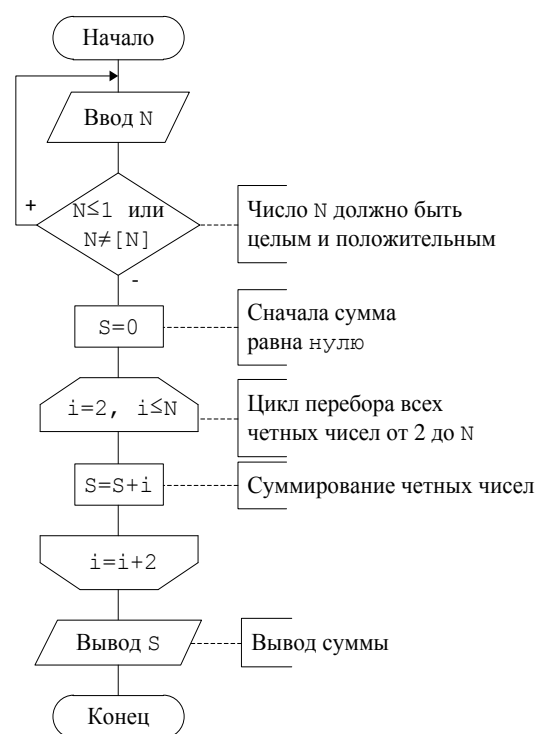


Рис. 1.29. «Более удачный» алгоритм вычисления суммы четных натуральных чисел цикла (рис. 1.29).

В таблице 1.7 приведены результаты тестирования алгоритма для **N=11**.

Таблица 1.7

Суммирование четных чисел

Шаг алгоритма	Проверка условия выхода из цикла $i \leq N$	S	i
До цикла		S=0	i= 2
1-й шаг цикла	$2 \leq 11$ ДА	$S=S+i= 0+ 2= 2$	$i= 2+2= 4$
2-й шаг цикла	$4 \leq 11$ ДА	$S=S+i= 2+ 4= 6$	$i= 4+2= 6$
3-й шаг цикла	$6 \leq 11$ ДА	$S=S+i= 6+ 6=12$	$i= 6+2= 8$
4-й шаг цикла	$8 \leq 11$ ДА	$S=S+i=12+ 8=20$	$i= 8+2=10$
5-й шаг цикла	$10 \leq 11$ ДА	$S=S+i=20+10=30$	$i=10+2=12$
Выход из цикла	$12 \leq 11$ НЕТ		
Результат		S=30	

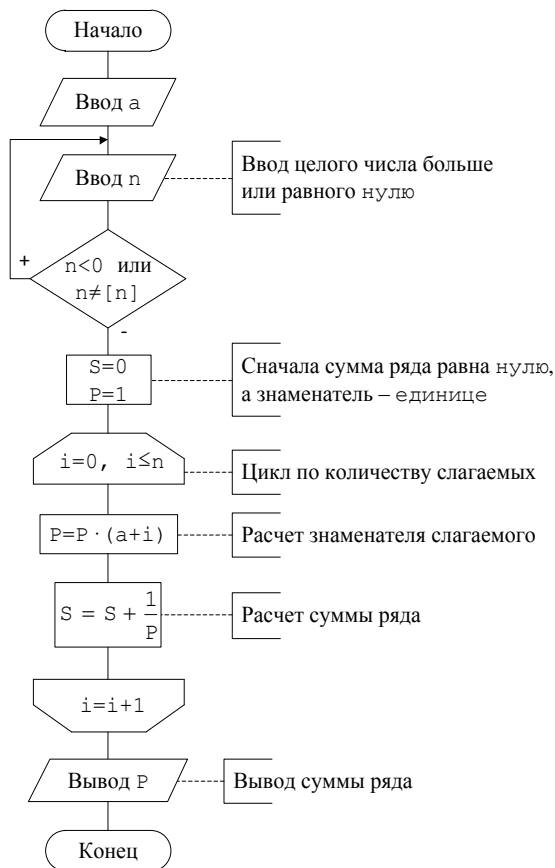


Рис. 1.30. Вычисление суммы ряда

Пример 15

Для заданных чисел вещественного **a** и целое положительное **n** **вычислить сумму ряда**:

$$\frac{1}{a} + \frac{1}{a \cdot (a+1)} + \dots + \frac{1}{a \cdot (a+1) \cdot \dots \cdot (a+n)}.$$

Дано (входные данные): **a** – вещественное число, **n** – целое положительное число.

Найти (выходные данные): **S** – сумму ряда, вещественное число.

Знаменатель каждого члена ряда вычисляется по формуле:

$$\prod_{j=0}^i a + j,$$

где **i=0, 1, ..., n** (номер элемента ряда).

Поэтому в данной задаче надо вычислять произведение (в знаменателе) и сумму дробей (рис. 1.30).

Выполним «ручную отладку» алгоритма, то есть проверим то, как будет работать алгоритм при каждом повторении цикла (табл. 1.8).

Таблица 1.8

Вычисление суммы ряда

Шаг алгоритма	i	Проверка условия выхода из цикла i ≤ n	P	S
До цикла			P=1	S=0
1-й шаг цикла	i=0	0 ≤ n ДА	P=P · (a+i) = =1 · (a+0) =a	S = S + $\frac{1}{P}$ = 0 + $\frac{1}{a}$ = $\frac{1}{a}$
2-й шаг цикла	i=1	1 ≤ n ДА	P= P · (a+i) = =a · (a+1)	S = S + $\frac{1}{P}$ = $\frac{1}{a}$ + $\frac{1}{a \cdot (a+1)}$
3-й шаг цикла	i=2	2 ≤ n ДА	P=P · (a+i) = =a · (a+1) · (a+2)	S = S + $\frac{1}{P}$ = $\frac{1}{a}$ + $\frac{1}{a \cdot (a+1)}$ + + $\frac{1}{a \cdot (a+1) \cdot (a+2)}$
...
n -й шаг цикла	i=n	n ≤ n ДА	P=P · (a+i) = =a · (a+1) · (a+2) · · (a+n)	S = S + $\frac{1}{P}$ = $\frac{1}{a}$ + $\frac{1}{a \cdot (a+1)}$ + + $\frac{1}{a \cdot (a+1) \cdot (a+2)}$ ++ $\frac{1}{a \cdot (a+1) \cdot (a+2) \cdot \dots \cdot (a+n)}$

Продолжение табл. 1.8

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	P	S
Выход из цикла	$i=n+1$	$n+1 \leq n$ НЕТ		
Результат				$\frac{1}{a} + \frac{1}{a \cdot (a+1)} +$ $+ \frac{1}{a \cdot (a+1) \cdot (a+2)} + \dots$ $+ \frac{1}{a \cdot (a+1) \cdot (a+2) \cdot \dots \cdot (a+n)}$

При решении задач, когда необходимо получить **приближенный результат с заданной точностью ε** (например, при вычислении сумм бесконечного ряда, при решении задач, реализуемых численными методами) получают алгоритмы с итерационными циклами.

Итерационный цикл – это цикл с неизвестным заранее числом повторений, окончание которого обычно осуществляется при достижении заданной точности вычислений.

Рассмотрим пример построения **итерационного цикла для вычисления суммы бесконечного ряда**.

Пример 16

Даны действительные x, ε ($x \neq 0, 0 < \varepsilon < 0,01$).

Вычислить с точностью ε сумму бесконечного ряда:

$$S = \sum_{k=0}^{\infty} \frac{(-1)^k \cdot x^k}{(2 \cdot k)!}.$$

Дано (входные данные): x – действительное число ($x \neq 0$), ε – действительное число, точность ($0 < \varepsilon < 0,01$).

Найти (выходные данные): S – сумму членов бесконечного ряда, вычисленную с точностью ε .

Распишем ряд:

$$S = 1 + \frac{-x}{2!} + \frac{x^2}{4!} + \frac{-1 \cdot x^3}{6!} + \dots + \frac{(-1)^n \cdot x^n}{(2 \cdot n)!} + \frac{(-1)^{n+1} \cdot x^{n+1}}{(2 \cdot n + 2)!} + \dots$$

$$k = 0 \quad 1 \quad 2 \quad 3 \quad \dots \quad n \quad n+1 \quad \dots$$

При числе k , стремящемся к бесконечности, члены данного ряда убывают по абсолютной величине. Значит, существует такой k -й член ряда, что все последующие члены ряда будут меньше ε по абсолютной величине и, следовательно, ими можно пренебречь при подсчёте суммы ряда с точностью ε . Чем меньше ε , тем больше количество шагов и тем точнее получается результат – большее количество цифр после запятой определяется точно. Отсюда следует, что можно составить алгоритм вычисления суммы ряда с точностью ε за конечное число шагов (итераций).

Для вычисления суммы ряда с точностью ε применим итерационный цикл, так как заранее нельзя определить, сколько членов ряда попадут в сумму.

Вычисления будем производить по следующему описанному в словесно-формульном виде алгоритму:

- 1) присвоить **k** значение **0**, **k** – указатель на номер члена ряда;
- 2) обнулить переменную для вычисления суммы:

$$S=0;$$

- 3) вычислить **k**-й член ряда по следующей формуле:

$$a_k = \frac{(-1)^k \cdot x^k}{(2 \cdot k)!};$$

- 4) добавить его значение к сумме:

$$S = S + a_k;$$

- 5) установить указатель на следующий член ряда:

$$k = k + 1;$$

- 6) если абсолютное значение члена ряда **a_k** больше или равно **ε**, то перейти к выполнению пункта 3;
- 7) распечатать сумму ряда **S** (**k** – количество суммированных членов ряда).

При рассмотрении данного алгоритма, можно обнаружить, что на третьем шаге приходится каждый раз вычислять факториал (в знаменателе) и возводить в степень число (в числителе) при вычислении очередного члена ряда.

На первый взгляд, может показаться, что придется организовать циклы для расчета факториала и степеней. При этом при увеличении **k** можно получить очень большие числа, при делении которых друг на друга произойдет потеря точности, поскольку количество значащих цифр, хранимых в ячейке памяти, ограничено. Кроме того, большие числа могут переполнить разрядную сетку.

Данную проблему можно обойти, если вычислять очередной член ряда не «с нуля», а умножением предыдущего члена ряда на некоторую величину – дополнение (**q_k**), которое определяется делением последующего члена ряда на предыдущий член ряда:

$$a_{k+1}=q_k \cdot a_k; \quad q_k=a_{k+1}/a_k;$$

$$\begin{aligned} q_k &= \frac{(-1)^{k+1} \cdot x^{k+1}}{(2 \cdot (k+1))!} \cdot \frac{(2 \cdot k)!}{(-1)^k \cdot x^k} = \frac{(-1)^k \cdot (-1) \cdot x^{k+1}}{(2 \cdot k + 2)!} \cdot \frac{(2 \cdot k)!}{(-1)^k \cdot x^k} = \\ &= \frac{(-1)^k \cdot (-1) \cdot x^k \cdot x}{(2 \cdot k)! \cdot (2 \cdot k + 1) \cdot (2 \cdot k + 2)} \cdot \frac{(2k)!}{(-1)^k \cdot x^k} = \frac{(-1) \cdot x}{(2 \cdot k + 1) \cdot (2 \cdot k + 2)} \end{aligned}$$

Учитывая дополнение **q_k** для вычисления суммы ряда необходимо вначале вычислить **a₀**, а затем в цикле на каждой итерации вычислять следующий член ряда, умножая предыдущий член ряда на соответствующее дополнение.

Для данного ряда **a₀=1** (подставили **k=0**).

Нулевой член ряда вычисляется по общей формуле при **k=0**, поэтому для каждого ряда **a₀** будет иметь своё значение (а не только **1**), которое и надо ему присвоить.

На каждой итерации требуется хранить значение одного члена ряда, поэтому все члены ряда формируются в одной и той же переменной **a**. Начальное значение переменной **a** совпадает со значением **a₀**.

Также в одной и той же переменной **S** будет накапливаться сумма. До цикла переменной **S** присваивается значение нулевого элемента ряда, а затем в цикле рассчитывается значение очередного элемента ряда и добавляется к сумме **S**.

Блок-схема алгоритма вычисления суммы бесконечного ряда приведена на рисунке 1.31.



Рис. 1.31. Вычисление суммы бесконечного ряда

Выполним «ручную отладку» алгоритма, то есть проверим то, как будет работать алгоритм при каждом повторении цикла (табл. 1.9).

Таблица 1.9

Вычисление суммы бесконечного ряда

Шаг алгоритма	a	k	S
До цикла	$a=1$	0	$S=1$
1-й шаг цикла	$a = 1 \cdot \frac{-1 \cdot x}{(2 \cdot 0 + 1) \cdot (2 \cdot 0 + 2)} = \frac{-x}{2}$	1	$S = 1 + \frac{-x}{2!}$

Продолжение табл. 1.9

Шаг алгоритма	a	k	S
2-й шаг цикла	$a = \frac{-x}{2} \cdot \frac{-1 \cdot x}{(2 \cdot 1 + 1) \cdot (2 \cdot 1 + 2)} =$ $= \frac{x^2}{2 \cdot 3 \cdot 4} = \frac{x^2}{4!}$	2	$S = 1 + \frac{-x}{2!} + \frac{x^2}{4!}$
3-й шаг цикла	$a = \frac{x^2}{2 \cdot 3 \cdot 4} \cdot \frac{-x}{(2 \cdot 2 + 1) \cdot (2 \cdot 2 + 2)} =$ $= \frac{-x^3}{2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} = \frac{-x^3}{6!}$	3	$S = 1 + \frac{-x}{2!} + \frac{x^2}{4!} + \frac{-x^3}{6!}$
...

Пример 17

Дано произвольное натуральное число **n**. *Определить все цифры числа и найти их сумму.*

Дано (входные данные): **n** – натуральное число.

Найти (выходные данные): **S** – сумму цифр числа.

В данной задаче, как и в предыдущей, также заранее невозможно определить, сколько раз повторятся действия по выделению цифр числа, и самим надо придумать условие, по которому прекращаются вычисления и осуществляется выход из цикла.

Применим итерационный цикл.

Можно предложить следующий алгоритм, описав его в словесно-формульном виде:

1) данное число **n** разделить на десять:

$$t = n / 10;$$

2) в полученном числе выделить целую часть:

$$P = [t];$$

3) найти разность полученного числа и его целой части, эту разность умножить на десять:

$$d = (t - P) \cdot 10.$$

Результат будет являться последней цифрой числа;

4) прибавить полученную цифру к сумме:

$$s = s + d;$$

5) проверить, не является ли значение **P** нулем. Если является, то это означает, что цифры все закончились и надо выходить из цикла. Если нет, то надо повторить действия, указанные в пунктах 1 – 4, предварительно присвоив **n** значение **P**. Так как в итерационном цикле всегда один из результатов, полученный в предыдущем повторении цикла (одно повторение называют итерацией), является исходным данным для следующей итерации.

Схема алгоритма приведена на рисунке 1.32. В этой схеме предусмотрена «защита от дурака» – первые два блока. Если будет введено $n \leq 0$, то ввод надо повторить.

В таблице 1.10 показана «ручная отладка» алгоритма для заданного числа **n=348**.

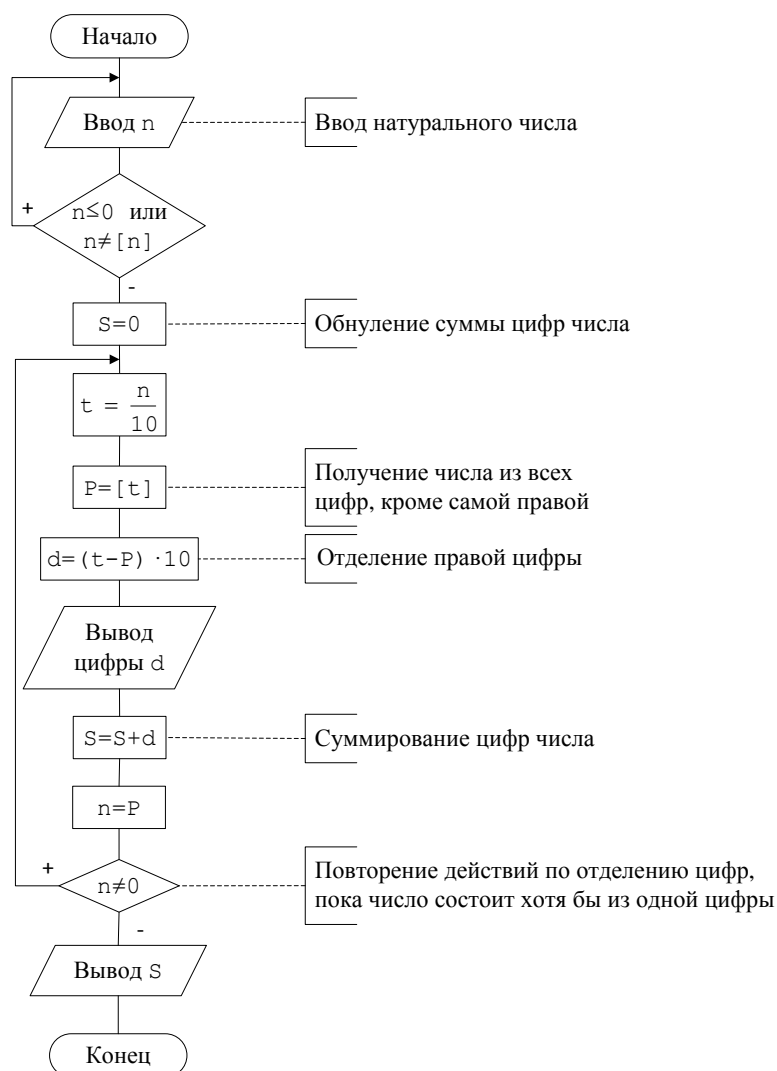


Рис. 1.32. Определение суммы цифр числа

Таблица 1.10

Определение суммы цифр числа **348**

Шаг алгоритма	t	P	d	S	n	Проверка условия выхода из цикла P ≠ 0
До цикла				S=0	n=348	
1-й шаг цикла	$t=n/10=348/10=34,8$	$P=[34,8]=34$	$d=(t-P) \cdot 10=(34,8-34) \cdot 10=0,8 \cdot 10=8$	$S=S+d=0+8=8$	n=34	$34 \neq 0$ ДА
2-й шаг цикла	$t=n/10=34/10=3,4$	$P=[3,4]=3$	$d=(t-P) \cdot 10=(3,4-3) \cdot 10=0,4 \cdot 10=4$	$S=S+d=8+4=12$	n=3	$3 \neq 0$ ДА
3-й шаг цикла	$t=n/10=3/10=0,3$	$P=[0,3]=0$	$d=(t-P) \cdot 10=(0,3-0) \cdot 10=0,3 \cdot 10=3$	$S=S+d=12+3=15$	n=0	$0 \neq 0$ НЕТ
Результат				S=15		

Пример 18

Дано натуральное число **N**. *Определить количество цифр в числе.*

Дано (входные данные): **N** – целое число больше **НУЛЯ**.

Найти (выходные данные): **kol** – количество цифр в числе.

Для того чтобы подсчитать количество цифр в числе, необходимо определить, сколько раз заданное число можно разделить на десять нацело (операция $\lfloor \cdot \rfloor$). Например, пусть **N=12345**, тогда количество цифр **kol=5**.

Это пример построения итерационного цикла, так как заранее не известно, сколько раз необходимо будет выполнять тело цикла (рис. 1.33).

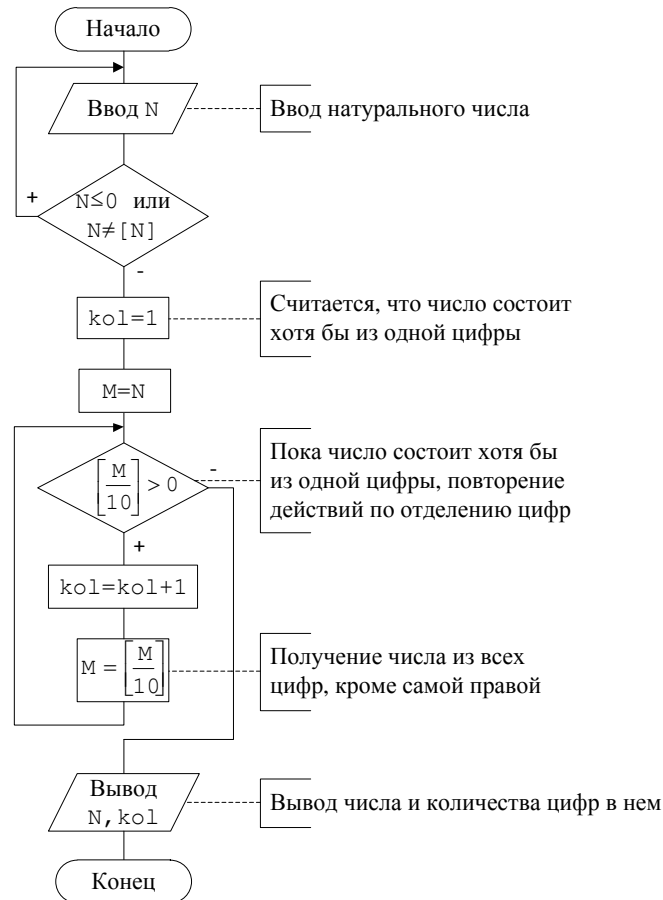


Рис. 1.33. Определение количества цифр в числе
(1-й вариант)

Промежуточные данные: **i** – параметр цикла, **M** – переменная для временного хранения значения **N**.

Результаты вычислений для чисел **12345** и **5** сведены в таблицы 1.11 и 1.12 соответственно.

На рисунке 1.33 представлен алгоритм с итерационным циклом с предусловием. Для решения этой же задачи можно составить алгоритм с итерационным циклом с постусловием (рис. 1.34). Проверка («ручная отладка») такого алгоритма показана в таблицах 1.13 и 1.14

Таблица 1.11

Определение количества цифр числа **12345**

Шаг алгоритма	Проверка условия выхода из цикла $[M/10] > 0$	kol	M
До цикла		kol=1	M=12345
1-й шаг цикла	$[12345/10]=1234 > 0$ ДА	kol=kol+1=2	M=[M/10]=[12345/10]=1234
Шаг алгоритма	Проверка условия выхода из цикла $[M/10] > 0$	kol	M
2-й шаг цикла	$[1234/10]=123 > 0$ ДА	kol=kol+1=3	M=[M/10]=[1234/10]=123
3-й шаг цикла	$[123/10]=12 > 0$ ДА	kol=kol+1=4	M=[M/10]=[123/10]=12
4-й шаг цикла	$[12/10]=1 > 0$ ДА	kol=kol+1=5	M=[M/10]=[12/10]=1
Выход из цикла	$[1/10]=0 > 0$ НЕТ		
Результат		kol=5	

Таблица 1.12

Определение количества цифр числа **5**

Шаг алгоритма	Проверка условия выхода из цикла	kol	M
До цикла		kol=1	M=5
Выход из цикла	$[5/10]=0 > 0$ НЕТ		
Результат		kol=1	

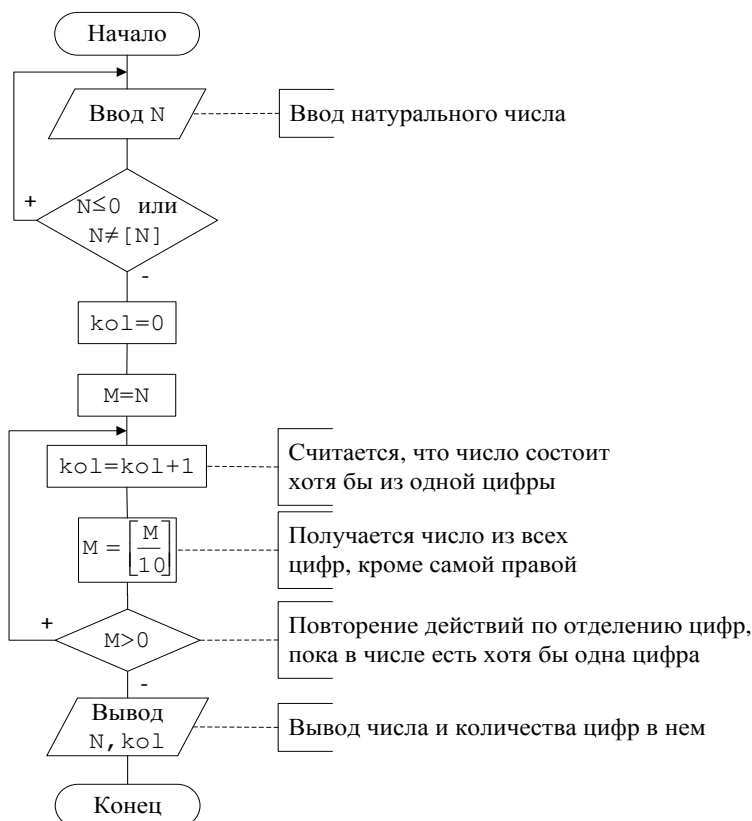


Рис. 1.34. Определение количества цифр в числе (2-й вариант)

Таблица 1.13

Определение количества цифр числа **12345**

Шаг алгоритма	kol	М	Проверка условия выхода из цикла М > 0
До цикла	kol=0	М=12345	
1-й шаг цикла	kol=kol+1=1	М=[М/10]=[12345/10]=1234	1234 > 0 ДА
2-й шаг цикла	kol=kol+1=2	М=[М/10]=[1234/10]=123	123 > 0 ДА
3-й шаг цикла	kol=kol+1=3	М=[М/10]=[123/10]=12	12 > 0 ДА
4-й шаг цикла	kol=kol+1=4	М=[М/10]=[12/10]=1	1 > 0 ДА
5-й шаг цикла	kol=kol+1=5	М=[М/10]=[1/10]=0	0 > 0 НЕТ
Результат	kol=5		

Таблица 1.14

Определение количества цифр числа **5**

Шаг алгоритма	kol	М	Проверка условия выхода из цикла М > 0
До цикла	kol=0	М=5	
1-й шаг цикла	kol=kol+1=1	М=[М/10]=[5/10]=0	0 > 0 НЕТ
Результат	kol=1		

Необходимо запомнить, что алгоритм записывается в виде блок-схемы без привязки к какому-либо языку программирования. Программист будет решать, как реализовать указанные в блок-схеме операции на конкретном языке, учитывая возможности языка программирования.

Например, в языке Си есть специальная *операция получения остатка от деления*, что упрощает вычисления, приведенные в рассмотренном примере. Кроме того, *при делении целых чисел* на языке Си, сразу получается целое число, то есть дробная часть автоматически отбрасывается. Что также упрощает вычисления.

1.7. РАБОТА С МАССИВАМИ

1.7.1. ПОНЯТИЕ МАССИВА

Многие задачи, которые решаются с использованием компьютера, связаны с обработкой больших объемов информации, представляющей совокупность данных, объединенных единым математическим содержанием или близких между собой по смыслу.

Примерами таких данных являются координаты, задающие положение точки в пространстве, коэффициенты системы линейных уравнений, значения некоторой функции в произвольных точках, коэффициенты многочлена и так далее. Эти данные удобно представлять в виде линейных или прямоугольных таблиц.

В математике табличные величины называются векторами или матрицами.

В алгоритме и программе для представления табличных данных используется понятие *массив*.

Массивом называется совокупность однотипных данных, имеющих одинаковое имя и отличающихся индексами.

Величины, составляющие массив, называются элементами массива.

Однотипность величин означает, что все элементы массива должны быть одного типа: числовые или символьные. Если элементы массива числа, то все числа также одного типа – целые или вещественные.

Любой элемент массива можно выделить с помощью индекса – порядкового номера элемента. Для этого указывается имя массива и индекс нужного элемента. Например: a_2 , a_7 .

В качестве индексов элементов могут использоваться целые числа, начиная от 0, переменные целого типа или выражения целого типа.

В разных языках программирования индекс записывается по-разному: в круглых скобках (QBASIC), в квадратных скобках (Си). Так как мы рассматриваем запись алгоритма в виде блок-схем, то индекс будем записывать, как принято в математике: нижним индексом.

Массив имеет три основные характеристики: **размерность, размер и длину**.

Размерность массива – это количество индексов у элементов. Это могут быть одно- и двумерные массивы (в QBASIC), большей размерности (в Си).

Размер массива – это количество элементов в массиве.

Длина массива – это количество байтов, занимаемых массивом в оперативной памяти ЭВМ. Она зависит от типа массива.

Пример одномерного массива

Пусть в программе необходимо произвести обработку последовательности из семи вещественных чисел.

Элементы массива: a_1 , ..., a_6 , a_7 .

В общем виде элементы массива записываются следующим образом:

$$a_i, \text{ где } i=1, 2, \dots, 7.$$

Это массив одномерный. Максимальное значение индекса – 7, поэтому транслятор выделяет в памяти $7 \cdot 4 = 28$ байт под весь массив.

Если счет индекса идет с **НУЛЯ**, то математическая запись элементов массива:

$$a_0, a_1, \dots, a_6.$$

В этом случае в общем виде элементы массива записываются следующим образом:

$$a_i, \text{ где } i=0, 1, \dots, 6.$$

Максимальное значение индекса – 6, но элементов 7, так как счет индекса идет с **НУЛЯ**.

QBASIC позволяет вести счет индексов как с **НУЛЯ**, так и с **ЕДИНИЦЫ**, вообще с любого числа. В языке Си счет индексов ведется строго с **НУЛЯ**.

Пример двумерного массива

Пусть в программе необходимо работать с таблицей целых чисел, содержащей две строки и три столбца:

$$\begin{pmatrix} 1 & 50 & -1 \\ 0 & 10 & 15 \end{pmatrix}$$

Такие данные надо рассматривать, как **двумерный** массив, так как местоположение каждого его элемента определяется двумя индексами. **Первый индекс – всегда номер строки, второй индекс – номер столбца**.

Если массиву дать имя **b** и счет индексов вести с единицы, то математическая запись таблицы будет такая:

$$b = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

В общем виде элементы массива записываются следующим образом:

$$b_{i,j}, \text{ где } i=1, 2, j=1, 2, 3.$$

Элементов $2 \cdot 3 = 6$, поэтому транслятор выделяет в памяти $6 \cdot 2 = 12$ байт под весь массив.

Пример ввода элементов одномерного массива

Перед выполнением непосредственных действий с элементами массива необходимо задать размер массива и его элементы.

Каждый элемент массива должен быть введен индивидуально в цикле по переменной, являющейся индексом элементов массива.

Типичный алгоритм ввода одномерного массива **a** размером **n** элементов приведен на рисунке 1.35.

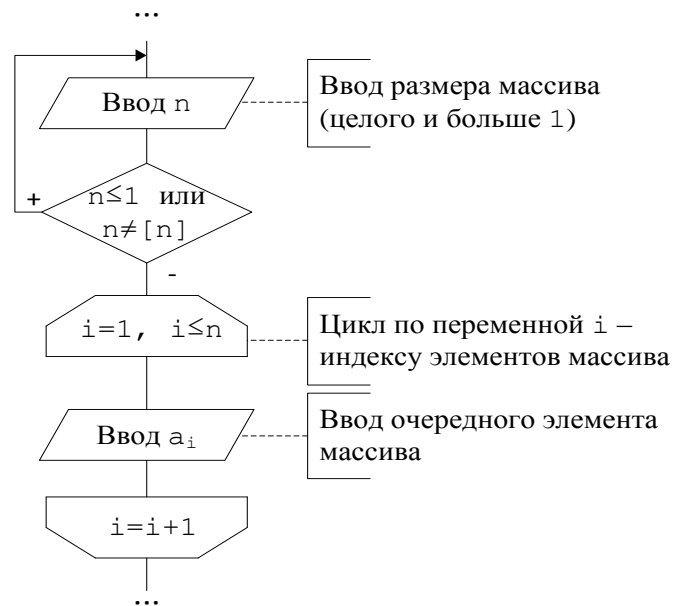


Рис. 1.35. Ввод одномерного массива

Пример вывода элементов одномерного массива

После ввода элементов массива для проверки ввода значений желательно вывести элементы массива.

После обработки массива необходимо вывести значения элементов массива. Причем не только в том случае, когда значения элементов преобразовывались, но даже в том случае, когда массив не изменялся. Это нужно для проверки того, что значения элементов случайно не изменились.

Аналогично вводу каждый элемент массива должен быть выведен индивидуально в цикле по переменной, являющейся индексом элементов массива.

Типичный алгоритм вывода одномерного массива **a** размером **n** элементов приведен на рисунке 1.36.

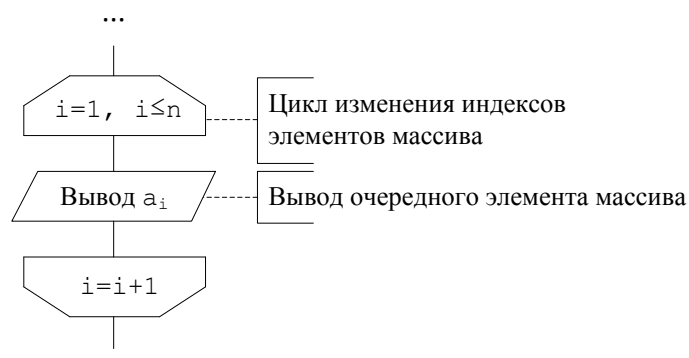


Рис. 1.36. Вывод одномерного массива

Пример ввода/вывода элементов двумерного массива

Матрицы, как и одномерные массивы, нужно вводить/выводить поэлементно. Блок-схема ввода элементов матрицы изображена на рисунке 1.37. Вывод матрицы организуется аналогично вводу (рисунок 1.38).

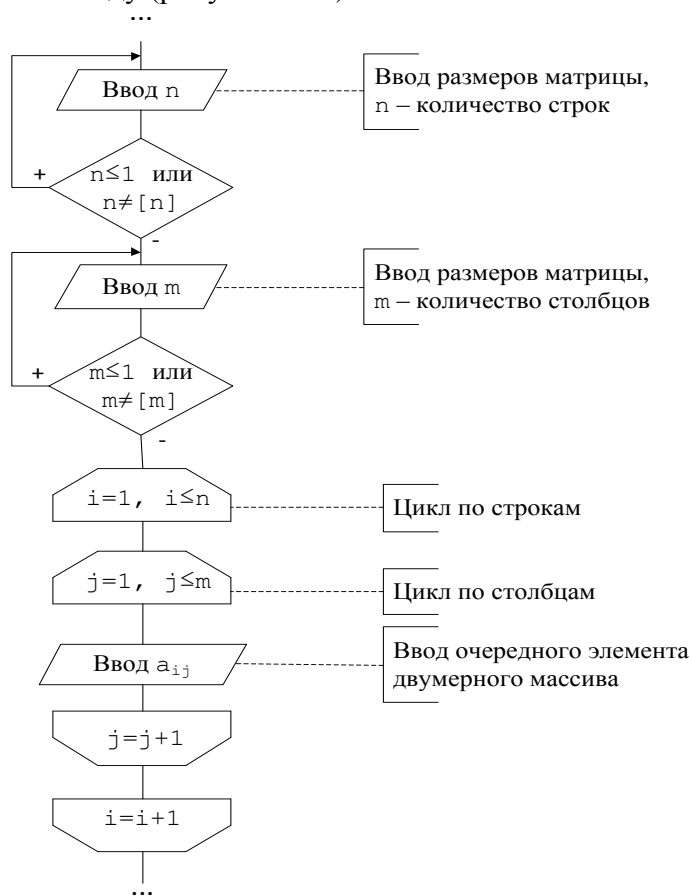


Рис. 1.37. Ввод матрицы

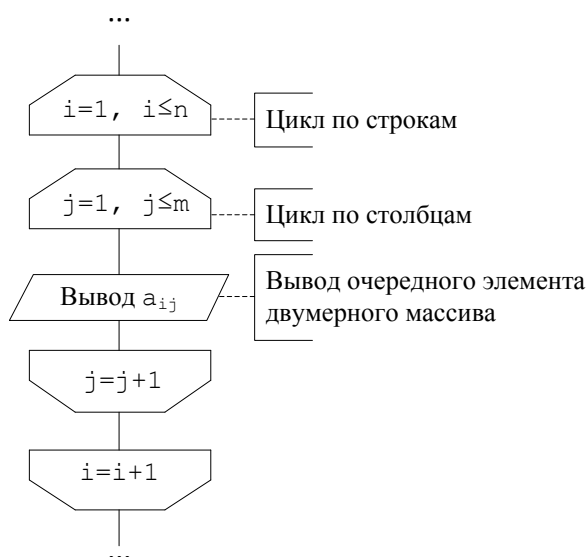


Рис. 1.38. Вывод матрицы

1.7.2. АЛГОРИТМЫ РАБОТЫ С ОДНОМЕРНЫМИ МАССИВАМИ

Рассмотрим несколько наиболее типичных алгоритмов работы с одномерными массивами.

Условимся во всех примерах *счет индексов вести с единицы*.

Пример 19

Задано n чисел. Необходимо *вычислить их сумму и произведение*.

Дано (входные данные): a_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): S – сумму, P – произведение элементов массива.

Алгоритм будет состоять из следующих шагов (рис. 1.39):

- ввод размера массива n ;
- ввод элементов массива по одному в цикле. Индекс i элемента массива будет выступать параметром цикла, и изменяться от 1 до n ;
- обращение в цикле к каждому элементу массива по индексу (перебор элементов массива) и накопление их суммы S и произведения P . Для перехода к последующему элементу a_i увеличивается значение номера элемента i на 1 ;
- вывод полученных значений суммы и произведения.

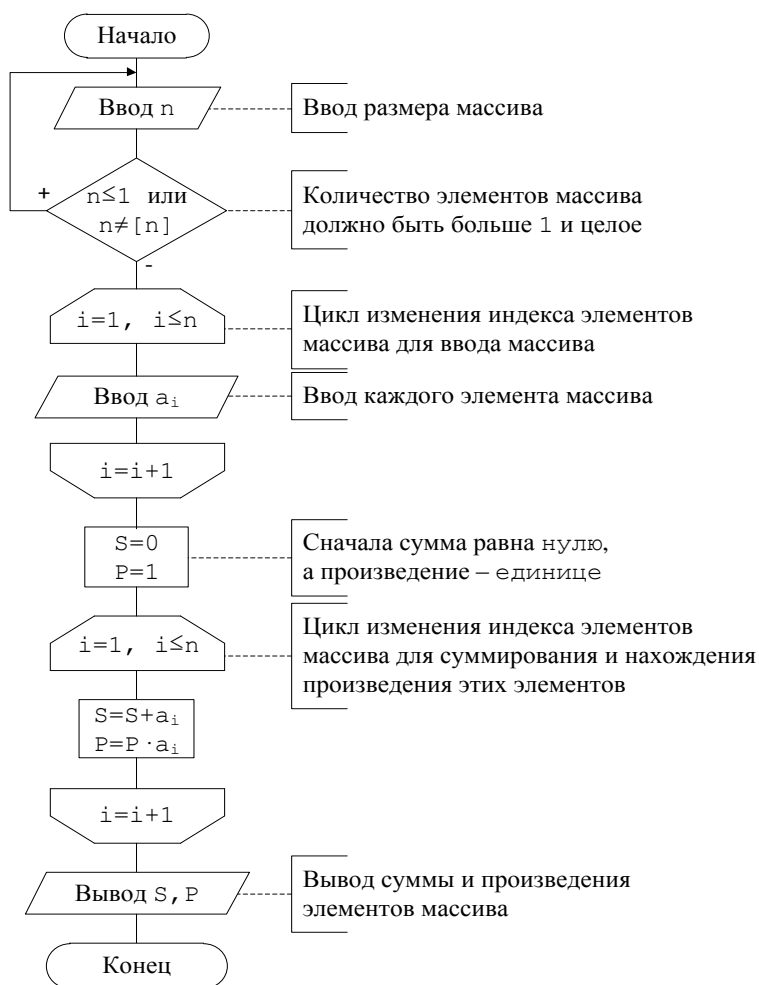


Рис. 1.39. Нахождение суммы и произведения элементов одномерного массива

Выполним «ручную отладку» алгоритма для массива из четырех элементов **5, 7, 8, 25**, то есть проверим, как будет работать алгоритм при каждом повторении цикла (табл. 1.15).

Таблица 1.15

Вычисление суммы и произведения элементов массива

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	a_i	S	P
1-й шаг цикла	$i=1$	$1 \leq 4$ ДА	$a_1= 5$		
2-й шаг цикла	$i=2$	$2 \leq 4$ ДА	$a_2= 7$		
3-й шаг цикла	$i=3$	$3 \leq 4$ ДА	$a_3= 8$		
4-й шаг цикла	$i=4$	$4 \leq 4$ ДА	$a_4=25$		
Выход из цикла	$i=5$	$5 \leq 4$ НЕТ			
До цикла				$S=0$	$P=1$
1-й шаг цикла	$i=1$	$1 \leq 4$ ДА		$S=S+a_1=$ $=0+5=5$	$P=P \cdot a_1=$ $=1 \cdot 5=5$
2-й шаг цикла	$i=2$	$2 \leq 4$ ДА		$S=S+a_2=$ $=5+7=12$	$P=P \cdot a_2=$ $=5 \cdot 7=35$
3-й шаг цикла	$i=3$	$3 \leq 4$ ДА		$S=S+a_3=$ $=12+8=20$	$P=P \cdot a_3=$ $=35 \cdot 8=280$
4-й шаг цикла	$i=4$	$4 \leq 4$ ДА		$S=S+a_4=$ $=20+25=45$	$P=P \cdot a_4=$ $=280 \cdot 25=7000$
Выход из цикла	$i=5$	$5 \leq 4$ НЕТ			
Результат				$S=45$	$P=32$

Пример 20

Пусть дан массив x из n вещественных чисел больших **НУЛЯ**. *Посчитать корень квадратный из произведения чисел, стоящих на четных местах*. Полученный результат записать на место первого элемента.

Дано (входные данные): x_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): $\sqrt{\prod_i x_i}$ для $i=2, 4, 6, \dots, n$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив положительных чисел:

Значение элемента	6	8	15	9	1	10	5	10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Результат должен иметь вид:

Значение элемента	120	8	15	9	1	10	5	10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

В алгоритме, приведенном на рисунке 1.40, не хватает «защиты от дурака» – проверки количества элементов n и вводимых чисел x_i на положительность.

Выполним «ручную отладку» алгоритма для массива из тестового примера, $n=10$ (табл. 1.16).

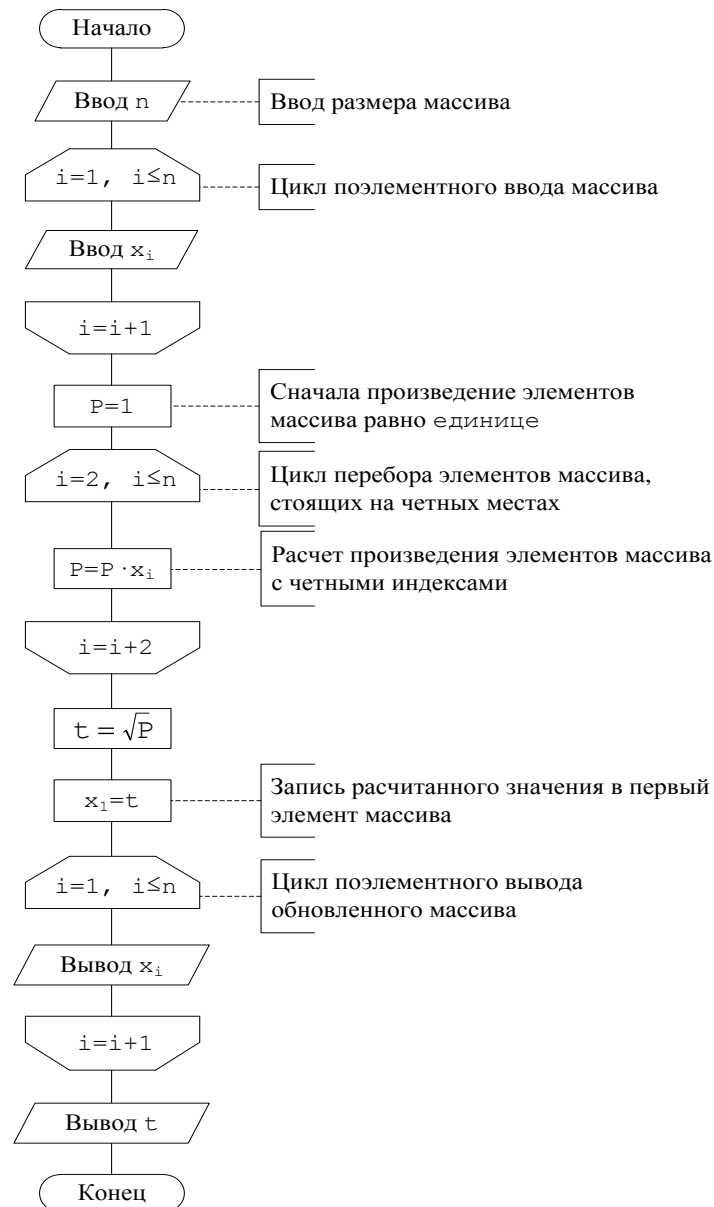


Рис. 1.40. Обработка элементов одномерного массива, стоящих на четных местах

Таблица 1.16

Вычисление корня квадратного из произведения элементов массива, стоящих на четных местах

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	x_i	P
1-й шаг цикла	i=1	$1 \leq 10$ ДА	$x_1 = 6$	
2-й шаг цикла	i=1+1=2	$2 \leq 10$ ДА	$x_2 = 8$	
3-й шаг цикла	i=2+1=3	$3 \leq 10$ ДА	$x_3 = 15$	
4-й шаг цикла	i=3+1=4	$4 \leq 10$ ДА	$x_4 = 9$	
5-й шаг цикла	i=4+1=5	$5 \leq 10$ ДА	$x_5 = 1$	
6-й шаг цикла	i=5+1=6	$6 \leq 10$ ДА	$x_6 = 10$	
7-й шаг цикла	i=6+1=7	$7 \leq 10$ ДА	$x_7 = 5$	
8-й шаг цикла	i=7+1=8	$8 \leq 10$ ДА	$x_8 = 10$	
9-й шаг цикла	i=8+1=9	$9 \leq 10$ ДА	$x_9 = 12$	

Продолжение табл. 1.16

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	x_i	P
10-й шаг цикла	$i=9+1=10$	$10 \leq 10$ ДА	$x_{10}=2$	
Выход из цикла	$i=10+1=11$	$11 \leq 10$ НЕТ		
До цикла				$P=1$
1-й шаг цикла	$i=2$	$2 \leq 10$ ДА		$P=P \cdot x_2=1 \cdot 8=8$
2-й шаг цикла	$i=2+2=4$	$4 \leq 10$ ДА		$P=P \cdot x_4=8 \cdot 9=72$
3-й шаг цикла	$i=4+2=6$	$6 \leq 10$ ДА		$P=P \cdot x_6=72 \cdot 10=720$
4-й шаг цикла	$i=6+2=8$	$8 \leq 10$ ДА		$P=P \cdot x_8=720 \cdot 10=7200$
5-й шаг цикла	$i=8+2=10$	$10 \leq 10$ ДА		$P=P \cdot x_{10}=7200 \cdot 2=14400$
Выход из цикла	$i=10+2=12$	$12 \leq 10$ НЕТ		
Результат			$x_1=120$	

Вопрос: Как изменить алгоритм, если надо найти корень квадратный из произведения чисел, стоящих *на нечетных местах*?

Ответ: Необходимо организовать цикл по переменной i также с шагом 2, но от 1 до n .

Вопрос: Как изменить алгоритм, если надо найти корень квадратный из произведения *только положительных* чисел, а массив состоит как из положительных, так и из отрицательных чисел?

Исходный массив любых чисел:

Значение элемента	6	-8	15	9	-1	0	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Результат должен иметь вид:

Значение элемента	312	-8	15	9	-1	0	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Ответ: Необходимо организовать цикл по переменной i с шагом 1, перебирать все элементы массива. В теле цикла перед вычислением произведения P необходимо поставить проверку того, положителен ли очередной элемент массива x_i , и вычислять произведение только при положительном результате проверки.

Пример 21

Пусть дан массив x из n вещественных чисел.

В данной последовательности x *найти максимальный элемент* и поменять его местами с последним элементом, если он сам не является максимальным. Если максимальных элементов несколько, то *взять последний максимальный элемент* по порядку в последовательности максимальных элементов.

Дано (входные данные): x_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): x_i , где $i=1, 2, \dots, n$, среди которых обменены два элемента, x_n равен максимальному элементу массива.

Составим тестовые примеры, чтобы более наглядно представить себе алгоритм:

1) Исходный массив с единственным максимальным элементом:

Значение элемента	6	-8	15	9	-1	0	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10
			max							

Результат должен иметь вид:

Значение элемента	6	-8	2	9	-1	0	5	-10	12	15
Индекс элемента	1	2	3	4	5	6	7	8	9	10

2) Исходный массив с несколькими максимальными элементами:

Значение элемента	6	-8	15	9	-1	15	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10
			max			max				

Результат должен иметь вид:

Значение элемента	6	-8	15	9	-1	2	5	-10	12	15
Индекс элемента	1	2	3	4	5	6	7	8	9	10

3) Исходный массив с последним максимальным элементом:

Значение элемента	6	-8	15	9	-1	15	5	-10	12	22
Индекс элемента	1	2	3	4	5	6	7	8	9	10
										max

Результат должен иметь вид:

Значение элемента	6	-8	15	9	-1	15	5	-10	12	22
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Существует несколько алгоритмов поиска максимального элемента.

Будем следовать следующему алгоритму (рис. 1.41):

1) сначала необходимо назначить некоторый **эталон** – переменную, которой заранее присваивается значение, например, первого элемента. Таким образом, примем за максимальный элемент (**х_м**) первый элемент, и запомним его номер в переменной **к=1**, так как нам надо не просто определить значение максимального элемента, но иметь возможность обменять значение максимального и последнего элементов. А для этого надо знать, на каком месте находится максимальный элемент, то есть какой у него индекс;

2) далее в цикле, изменяя **i** от **2** до **n**, будем сравнивать последовательно элементы **х_i** с содержимым эталона – переменной **х_м**. Если встретится элемент **х_i ≥ х_м**, то будем запоминать его в переменной **х_м**, а в переменной **к** – его номер. При выходе из цикла в **х_м** окажется **самый большой элемент** последовательности, а в **к** – его номер (**номер самого последнего самого большого элемента**);

3) останется поменять местами элемент с номером **к** и последний элемент с номером **n**, то есть **х_к** и **х_n**, но только в том случае, если последний элемент **х_n** не является максимальным **х_к**.

Вопрос: Как изменить алгоритм, если для обмена местами с последним элементом последовательности **брать первый по порядку максимальный элемент** в последовательности максимальных элементов?

Исходный массив с несколькими максимальными элементами:

Значение элемента	6	-8	15	9	-1	15	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10
			max			max				

Результат должен иметь вид:

Значение элемента	6	-8	2	9	-1	15	5	-10	12	15
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Ответ: В теле цикла изменить условие на строгое, тогда при выходе из цикла переменная **к** будет иметь значение индекса первого самого большого элемента массива.



Рис. 1.41. Поиск максимального элемента в массиве

Вопрос: Как изменить алгоритм, если *менять* местами максимальный элемент с *первым элементом* последовательности?

Ответ: Нужно поменять местами элемент с номером k и элемент с номером 1 , то есть x_k и x_1 .

Вопрос: Как изменить алгоритм, если *находить минимальный элемент* массива?

Ответ: В теле цикла изменить условие на \leq , то есть искать не *большой* элемент, а *меньший*, чем значение эталона – переменной xm . Тогда при выходе из цикла переменная k будет иметь значение индекса самого последнего самого наименьшего элемента массива.

Выполним «ручную отладку» алгоритма для массива с несколькими максимальными элементами из второго тестового примера, $n=10$ (табл. 1.17).

Таблица 1.17

Поиск последнего максимального элемента в массиве

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	Сравнение значений элементов массива со значением эталона $x_i \geq xm$	Значение эталона xm , k – номер максимального элемента	x_i
1-й шаг цикла	$i=1$	$1 \leq 10$ ДА			$x_1=6$
2-й шаг цикла	$i=1+1=2$	$2 \leq 10$ ДА			$x_2=-8$
3-й шаг цикла	$i=2+1=3$	$3 \leq 10$ ДА			$x_3=15$
4-й шаг цикла	$i=3+1=4$	$4 \leq 10$ ДА			$x_4=9$
5-й шаг цикла	$i=4+1=5$	$5 \leq 10$ ДА			$x_5=-1$
6-й шаг цикла	$i=5+1=6$	$6 \leq 10$ ДА			$x_6=15$
7-й шаг цикла	$i=6+1=7$	$7 \leq 10$ ДА			$x_7=5$
8-й шаг цикла	$i=7+1=8$	$8 \leq 10$ ДА			$x_8=-10$
9-й шаг цикла	$i=8+1=9$	$9 \leq 10$ ДА			$x_9=12$
10-й шаг цикла	$i=9+1=10$	$10 \leq 10$ ДА			$x_{10}=2$
Выход из цикла	$i=10+1=11$	$11 \leq 10$ НЕТ			
До цикла				$xm=x_1=6$ $k=1$	
1-й шаг цикла	$i=2$	$2 \leq 10$ ДА	$x_2=-8 \geq 6$ НЕТ		
2-й шаг цикла	$i=2+1=3$	$3 \leq 10$ ДА	$x_3=15 \geq 6$ ДА	$xm=x_3=15$ $k=3$	
3-й шаг цикла	$i=3+1=4$	$4 \leq 10$ ДА	$x_4=9 \geq 15$ НЕТ		
4-й шаг цикла	$i=4+1=5$	$5 \leq 10$ ДА	$x_5=-1 \geq 15$ НЕТ		
5-й шаг цикла	$i=5+1=6$	$6 \leq 10$ ДА	$x_6=15 \geq 15$ ДА	$xm=x_3=15$ $k=6$	
6-й шаг цикла	$i=6+1=7$	$7 \leq 10$ ДА	$x_7=5 \geq 15$ НЕТ		
7-й шаг цикла	$i=7+1=8$	$8 \leq 10$ ДА	$x_8=-10 \geq 15$ НЕТ		
8-й шаг цикла	$i=8+1=9$	$9 \leq 10$ ДА	$x_9=12 \geq 15$ НЕТ		
9-й шаг цикла	$i=9+1=10$	$10 \leq 10$ ДА	$x_{10}=2 \geq 15$ НЕТ		
Выход из цикла	$i=10+1=11$	$11 \leq 10$ НЕТ			
Результат	$xm=15, k=6$				

Пример 22

Даны два целочисленных массива a и b одинакового размера n .

Получить новый массив c , который состоит из элементов данных массивов a и b , расположенных следующим образом: $a_1, b_1, a_2, b_2, \dots, a_n, b_n$.

Дано (входные данные): a_i, b_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): c_i , где $i=1, 2, \dots, 2 \cdot n$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив **a**:

Значение элемента	6	-8	15	9	-1
Индекс элемента	1	2	3	4	5

Исходный массив **b**:

Значение элемента	0	5	-10	12	2
Индекс элемента	1	2	3	4	5

Результирующий массив **c** должен иметь вид:

Значение элемента	6	0	-8	5	15	-10	9	12	-1	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Должно быть такое соответствие элементов:

a ₁	b ₁	a ₂	b ₂	a ₃	b ₃	...	a _n	b _n
c ₁	c ₂	c ₃	c ₄	c ₅	c ₆	...	c _{2n-1}	c _{2n}

То есть элементы массива **c** с нечетными номерами – это элементы массива **a**, а элементы массива **c** с четными номерами – это элементы массива **b**.

Если изменять счетчик **i** индексов исходных массивов от **1** до **n**, то получим

- при **i=1**:

$$2 \cdot i - 1 = 2 \cdot 1 - 1 = 1, 2 \cdot i = 2 \cdot 1 = 2$$

– значения индексов для первых двух элементов нового массива **c**;

- при **i=2**

$$2 \cdot i - 1 = 2 \cdot 2 - 1 = 3, 2 \cdot i = 2 \cdot 2 = 4$$

– значения индексов для следующих двух элементов нового массива **c**.

Алгоритм приведен на рисунке 1.42.

Пример 23

Дан массив **a** размера **n**. *Расположить элементы заданного массива в обратном порядке.*

Дано (входные данные): **a_i**, где **i=1, 2, ..., n**.

Найти (выходные данные): **a_i**, где **i=1, 2, ..., n**.

Составим тестовые примеры, чтобы более наглядно представить себе алгоритм.

Исходный массив из четного числа элементов:

Значение элемента	6	-8	15	9	-1	0	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10
Индекс элемента для обмена	10	9	8	7	6	5	4	3	2	1
Номер пары для обмена	1	2	3	4	5	5	4	3	2	1

Результат должен иметь вид:

Значение элемента	2	12	-10	5	0	-1	9	15	-8	6
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Исходный массив из нечетного числа элементов:

Значение элемента	6	-8	15	9	-1	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9
Индекс элемента для обмена	9	8	7	6	-	4	3	2	1
Номер пары для обмена	1	2	3	4	Остается на месте	4	3	2	1

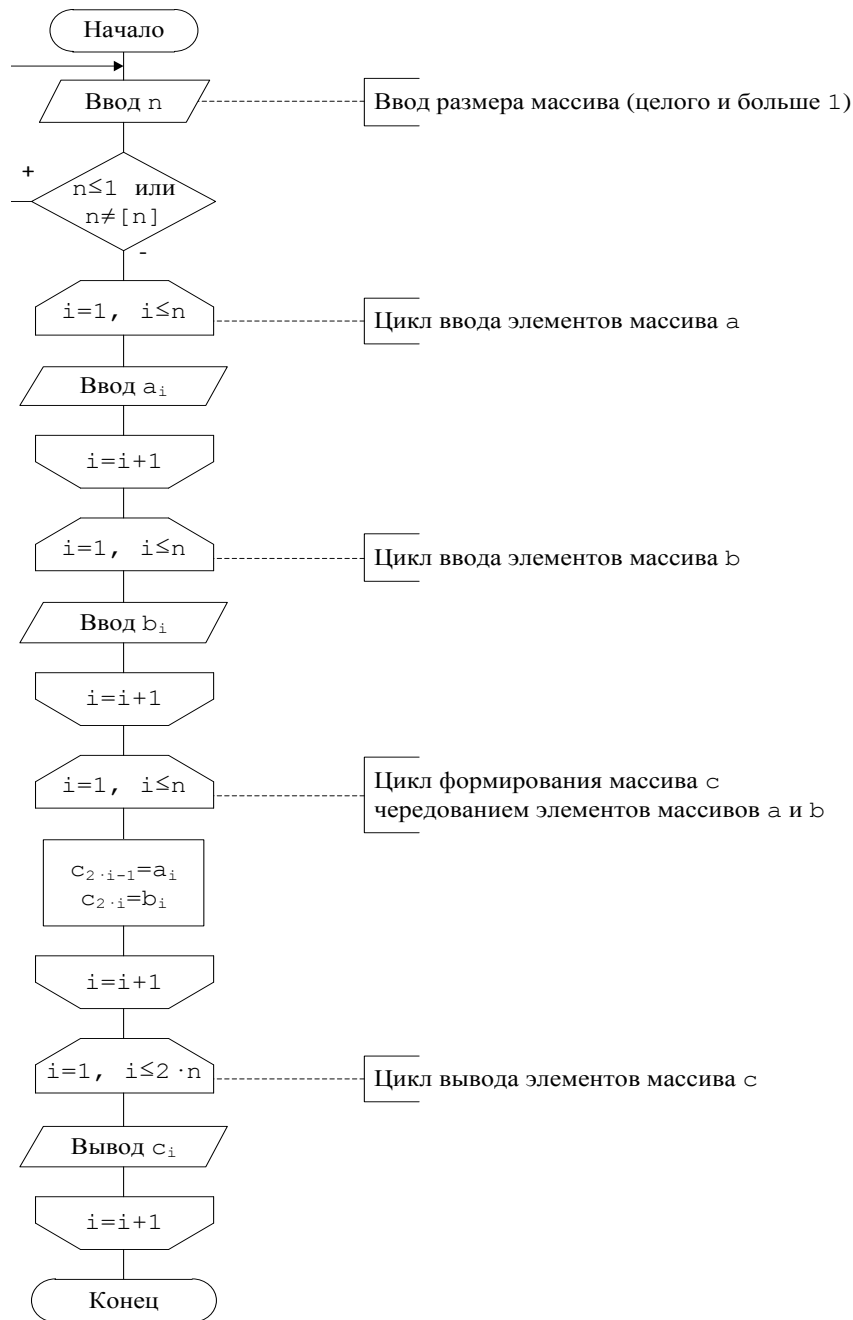


Рис. 1.42. Объединение двух массивов чередованием элементов

Результат должен иметь вид:

Значение элемента	2	12	-10	5	-1	9	15	-8	6
Индекс элемента	1	2	3	4	5	6	7	8	9

Нужно организовать цикл перебора элементов массива до середины, то есть $i=1, 2, 3, \dots, [n/2]$ и менять попарно значения 1 -го и n -го элементов, затем 2 -го и $n-1$ -го элементов, затем 3 -го и $n-2$ -го элементов, и так далее до середины массива (значение элемента с номером i следует обменять со значением элемента с номером $n-i+1$).

Блок-схема представлена на рисунке 1.43.

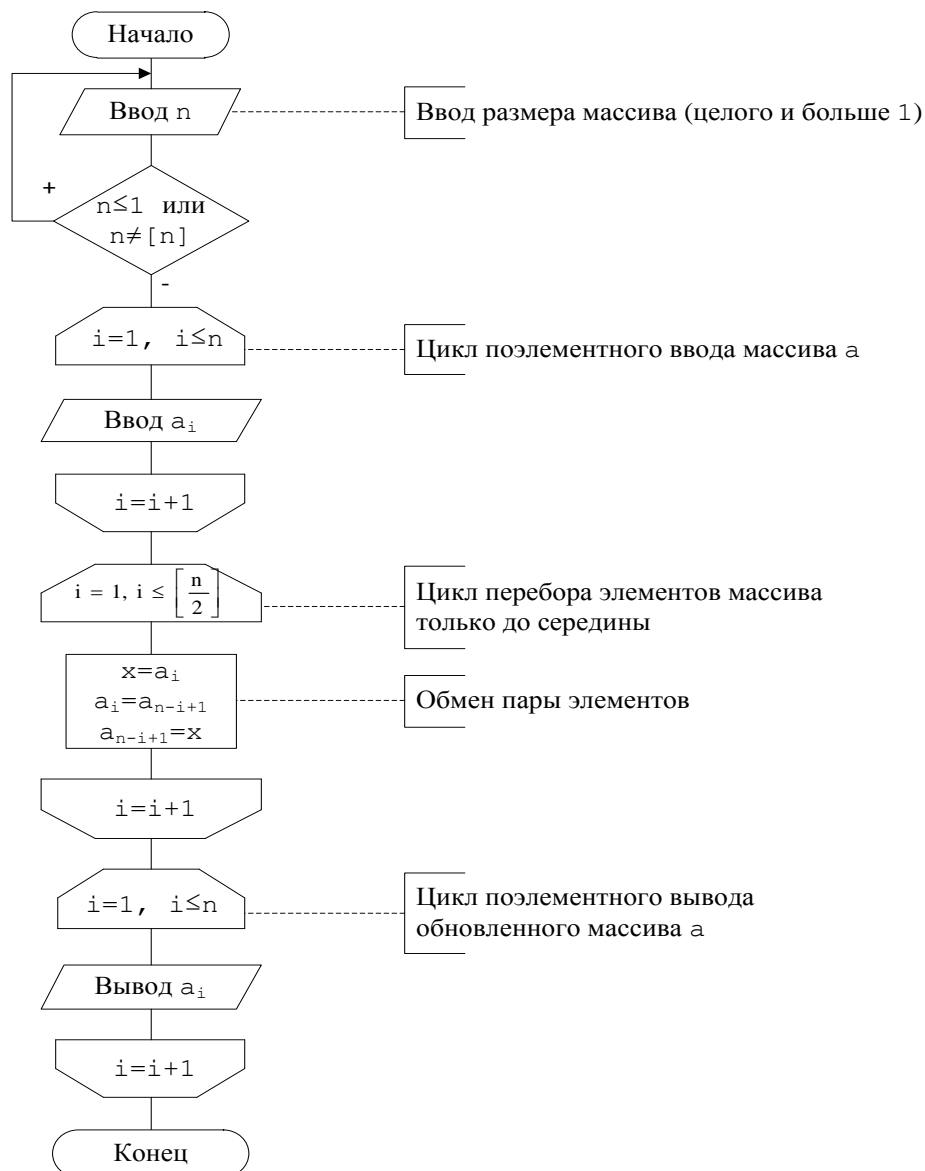


Рис. 1.43. Расположение элементов массива в обратном порядке

Пример 24

Дан массив **a** размера **n**.

Изменить массив, удалив из него отрицательные элементы, а значения положительных элементов увеличить на 1. **Определить количество элементов в новом массиве**.

Дано (входные данные): a_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): $a_i \geq 0$, где $i=1, 2, \dots, m$; **m** – количество элементов.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив:

Значение элемента	6	-8	15	9	-1	3	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Результат удаления отрицательных элементов должен иметь вид:

Значение элемента	6	15	9	3	5	12	2
Индекс элемента	1	2	3	4	5	6	7

Конечный результат после увеличения значений на 1 должен иметь вид:

Значение элемента	7	16	10	4	6	13	3
Индекс элемента	1	2	3	4	5	6	7

Проще всего решать эту задачу с использованием дополнительного массива. В этом случае при просмотре исходного массива элементы, которые требуется оставить, помещаются один за другим во второй массив. Однако для массивов больших размеров выделение двойного объема компьютерной памяти может оказаться слишком расточительным. Поэтому будем переформировывать исходный массив за счет сдвига влево положительных элементов, тем самым затирая отрицательные элементы и укорачивая массив.

Для этого используем следующий алгоритм (рис. 1.44).

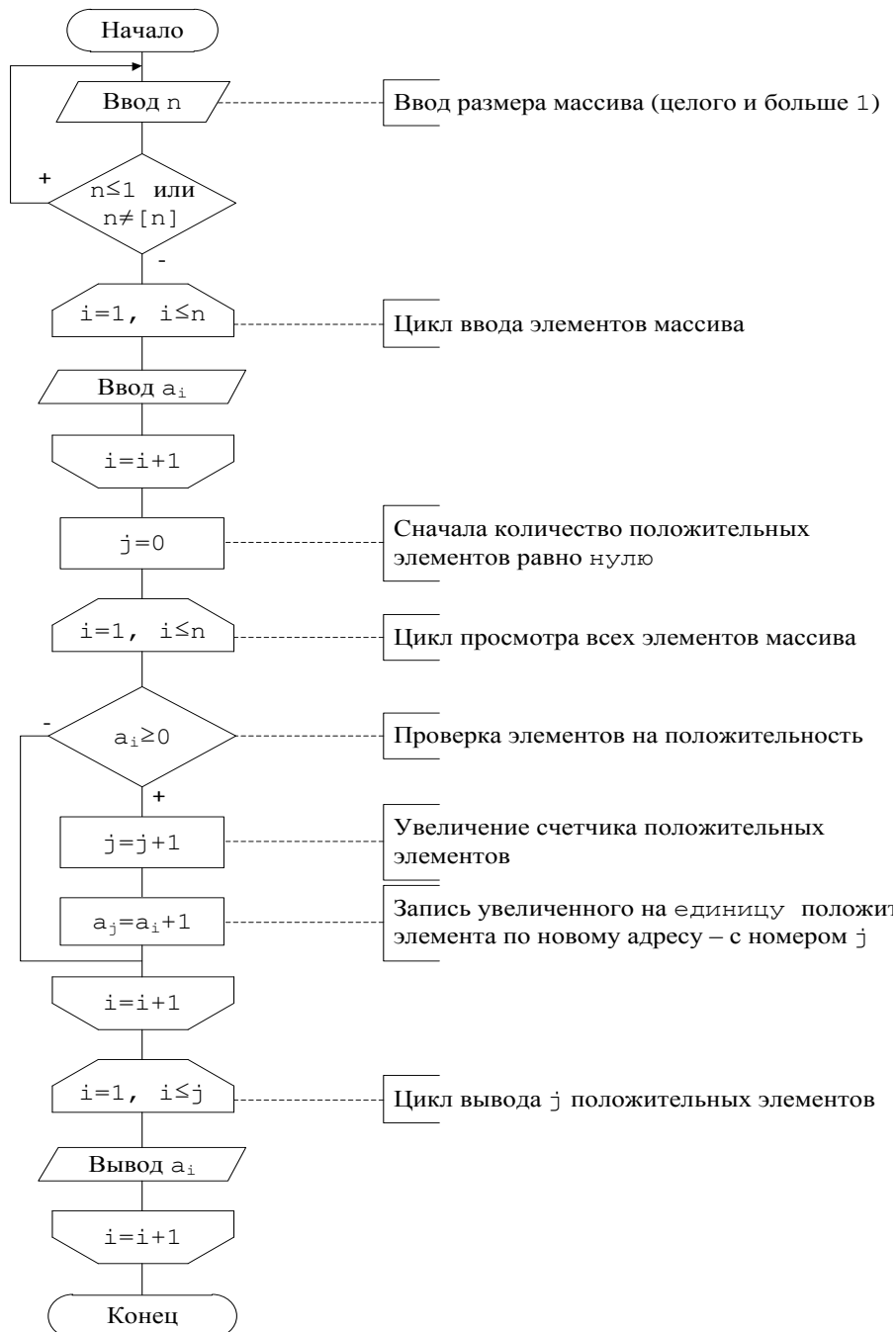


Рис. 1.44. Удаление элементов из массива

Установим начальное значение индекса обновленного массива **a** как **j=0**, то есть сначала в обновленном массиве ноль элементов.

Организуем цикл по переменной **i** от **1** до **n**, и будем просматривать каждый элемент массива **a_i**, и сравнивать его с нулем.

Если встретится отрицательное число, то будем его «пропускать» и переходить к следующему элементу массива.

Если встретится положительное число, то будем изменять его значение по формуле **a_i=a_i+1**, будем увеличивать индекс обновленного массива **j=j+1**, и будем переписывать значение этого элемента на новое место с номером **j**.

В результате просмотра элементов всего массива будет сформирован новый массив с количеством элементов, равным значению переменной **j**.

Выполним «ручную отладку» алгоритма для массива из тестового примера, **n=10** (табл. 1.18).

Таблица 1.18

Удаление из массива отрицательных элементов

Шаг алгоритма	i	Проверка условия выхода из цикла i ≤ n	Проверка числа на положительность a_i ≥ 0	j	a_i
1-й шаг цикла	i=1	1 ≤ 10 ДА			a ₁ = 6
2-й шаг цикла	i=1+1=2	2 ≤ 10 ДА			a ₂ = -8
3-й шаг цикла	i=2+1=3	3 ≤ 10 ДА			a ₃ = 15
4-й шаг цикла	i=3+1=4	4 ≤ 10 ДА			a ₄ = 9
5-й шаг цикла	i=4+1=5	5 ≤ 10 ДА			a ₅ = -1
6-й шаг цикла	i=5+1=6	6 ≤ 10 ДА			a ₆ = 3
7-й шаг цикла	i=6+1=7	7 ≤ 10 ДА			a ₇ = 5
8-й шаг цикла	i=7+1=8	8 ≤ 10 ДА			a ₈ =-10
9-й шаг цикла	i=8+1=9	9 ≤ 10 ДА			a ₉ = 12
10-й шаг цикла	i=9+1=10	10 ≤ 10 ДА			a ₁₀ = 2
Выход из цикла	i=10+1=11	11 ≤ 10 НЕТ			
До цикла				j=0	
1-й шаг цикла	i=1	1 ≤ 10 ДА	a ₁ =6 ≥ 0 ДА	j=j+1= =0+1=1	a _j =a _i +1 a₁ =a ₁ +1= =6+1=7
2-й шаг цикла	i=1+1=2	2 ≤ 10 ДА	a ₂ =-8 ≥ 0 НЕТ		
3-й шаг цикла	i=2+1=3	3 ≤ 10 ДА	a ₃ =15 ≥ 0 ДА	j=j+1= =1+1=2	a _j =a _i +1 a₂ =a ₃ +1= =15+1=16
4-й шаг цикла	i=3+1=4	4 ≤ 10 ДА	a ₄ =9 ≥ 0 ДА	j=j+1= =2+1=3	a _j =a _i +1 a₃ =a ₄ +1= =9+1=10
5-й шаг цикла	i=4+1=5	5 ≤ 10 ДА	a ₅ =-1 ≥ 0 НЕТ		
6-й шаг цикла	i=5+1=6	6 ≤ 10 ДА	a ₆ =3 ≥ 0 ДА	j=j+1= =3+1=4	a _j =a _i +1 a₄ =a ₆ +1= =3+1=4
7-й шаг цикла	i=6+1=7	7 ≤ 10 ДА	a ₇ =5 ≥ 0 ДА	j=j+1= =4+1=5	a _j =a _i +1 a₅ =a ₇ +1= =5+1=6
8-й шаг цикла	i=7+1=8	8 ≤ 10 ДА	a ₈ =-10 ≥ 0 НЕТ		
9-й шаг цикла	i=8+1=9	9 ≤ 10 ДА	a ₉ =12 ≥ 0 ДА	j=j+1= =5+1=6	a _j =a _i +1 a₆ =a ₉ +1= =12+1=13
10-й шаг цикла	i=9+1=10	10 ≤ 10 ДА	a ₁₀ =2 ≥ 0 ДА	j=j+1= =6+1=7	a _j =a _i +1 a₇ =a ₁₀ +1 =2+1=3
Выход из цикла	i=10+1=11	11 ≤ 10 НЕТ			
Результат	j=7 положительных элементов				

Пример 25

Задан массив из n элементов. *Сформировать массивы номеров* положительных и отрицательных элементов.

Дано (входные данные): a_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): b_i – индексы положительных элементов, где $i=1, 2, \dots, m$; c_i – индексы отрицательных элементов, где $i=1, 2, \dots, k$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив:

Значение элемента	6	-8	15	9	-1	3	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Результатом будут два новых массива следующего вида:

1) массив индексов положительных элементов исходного массива

Значение элемента	1	3	4	6	7	9	10
Индекс элемента	1	2	3	4	5	6	7

2) массив индексов отрицательных элементов исходного массива

Значение элемента	2	5	8
Индекс элемента	1	2	3

Блок-схема формирования массивов номеров элементов представлена на рис. 1.45.

Пример 26

Даны натуральное число n , действительные числа a_1, \dots, a_n . *Преобразовать последовательность* a_1, \dots, a_n , расположив вначале отрицательные числа, а затем – положительные (нулевые элементы считать положительными). При этом порядок как отрицательных, так и неотрицательных чисел сохранить прежним.

Дано (входные данные): a_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): $a_i < 0$, где $i=1, 2, \dots, k$, $a_i \geq 0$, где $i=k+1, k+2, \dots, n$, k – количество отрицательных элементов массива.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив:

Значение элемента	6	-8	15	9	-1	0	5	-10	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

Результат должен иметь вид:

Значение элемента	-8	-1	-10	6	15	9	0	5	12	2
Индекс элемента	1	2	3	4	5	6	7	8	9	10

В целях экономии компьютерной памяти *не будем формировать два дополнительных массива* с положительными элементами и с отрицательными элементами, чтобы затем уже их элементы переписывать в результирующий массив.

Будем переформировывать исходный массив. Алгоритм представлен на рисунке 1.46. Переменная L – индекс отрицательных элементов в начале массива. Изменяя переменную i от 1 до n , перебираем по очереди все элементы массива a_i . Как только встречается отрицательное число, оно запоминается во вспомогательной переменной d , а все положительные числа слева от него, сдвигаются на одну позицию вправо, чтобы освободить место для найденного отрицательного числа. После сдвига группы положительных чисел, слева от них на позицию L записывается найденное отрицательное число, запомненное во вспомогательной переменной d . Затем индекс отрицательных элементов увеличивается на 1.

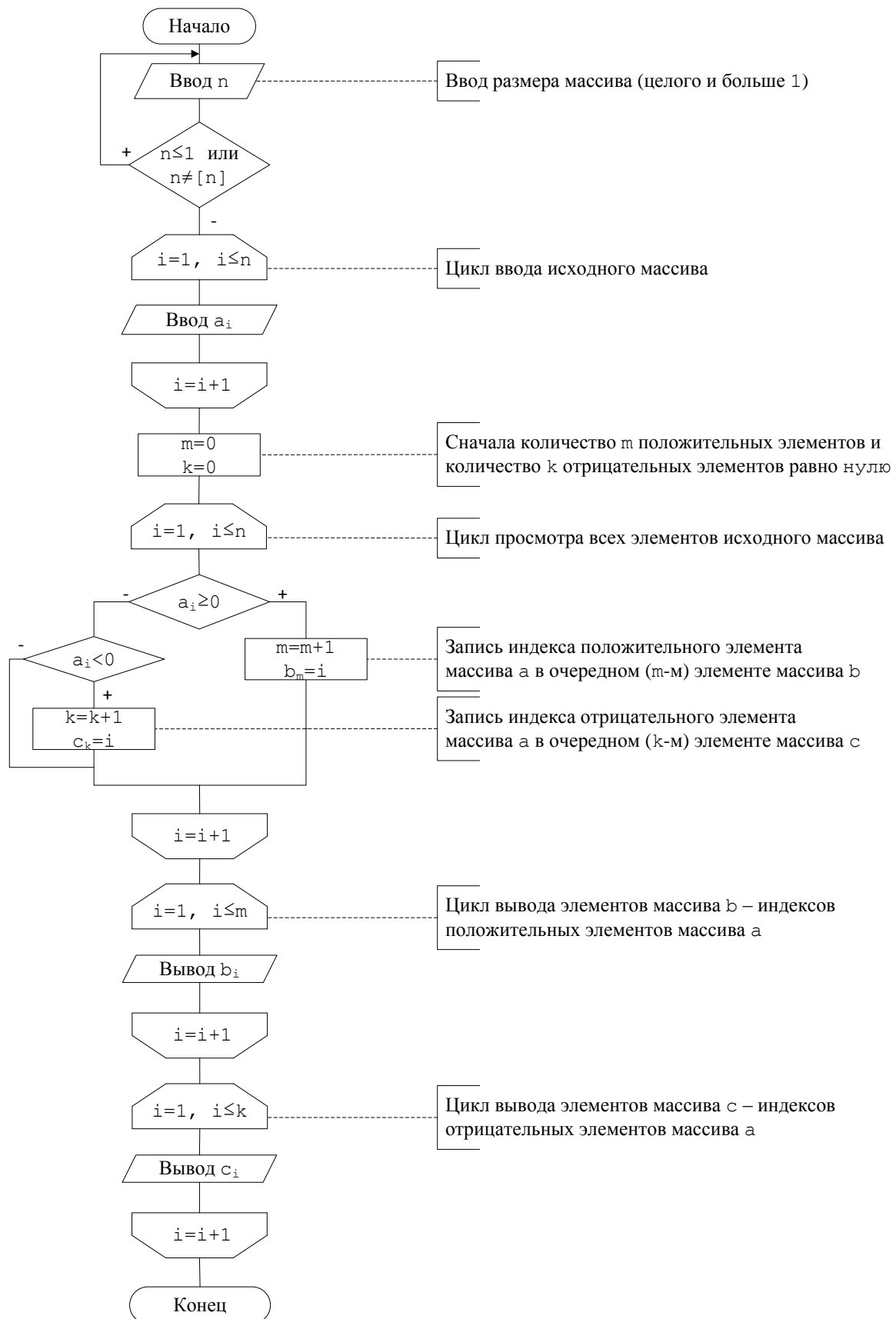


Рис. 1.45. Формирование массивов номеров положительных и отрицательных элементов одномерного массива

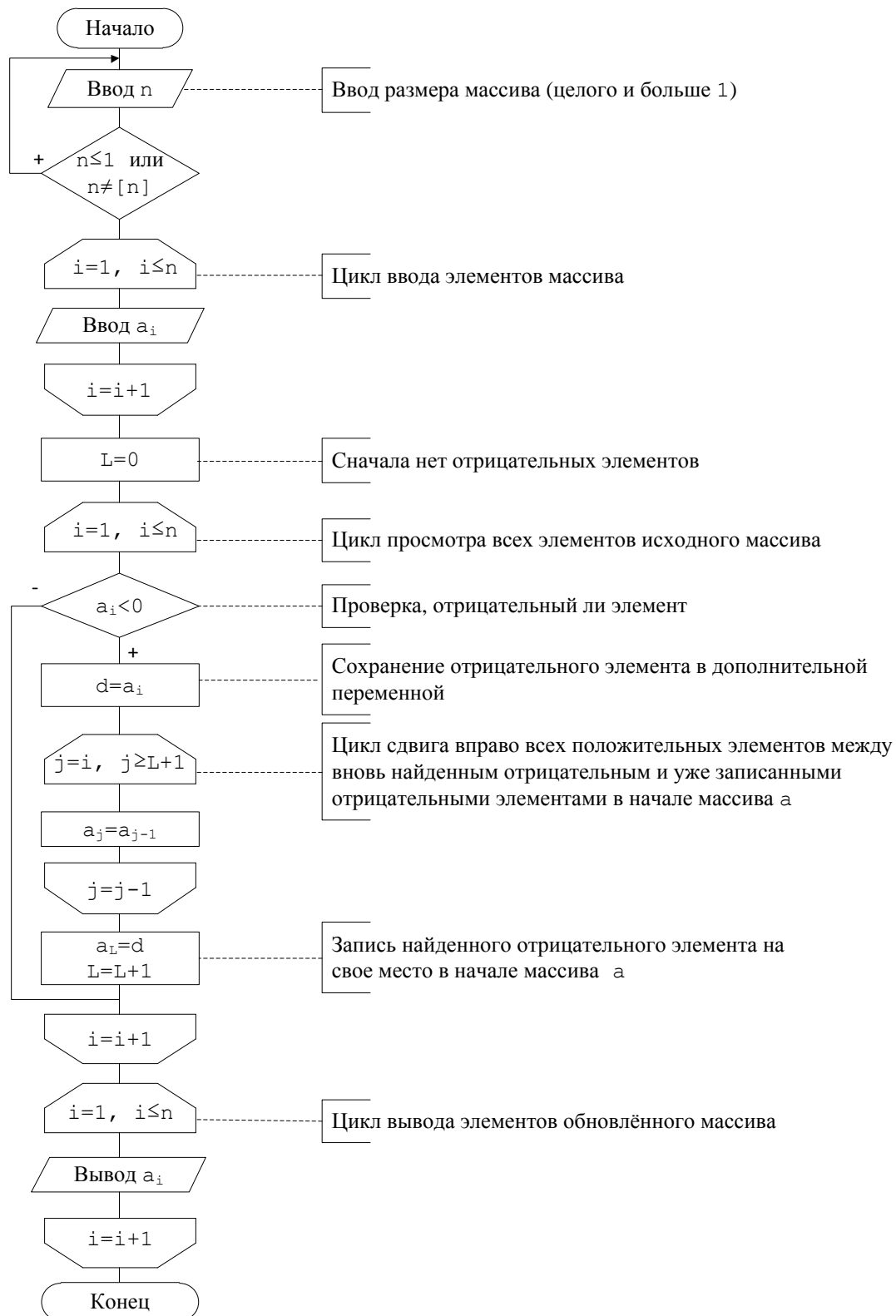


Рис. 1.46. Преобразование одномерного массива – расположение в массиве сначала отрицательных, затем положительных элементов

Выполним «ручную отладку» алгоритма для массива из тестового примера, $n=10$ (табл. 1.19).

Таблица 1.19

Расположение в массиве сначала отрицательных, затем положительных элементов

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	Проверка числа $a_i < 0$	d	a_i	L
1-й шаг цикла	$i=1$	$1 \leq 10$ ДА			$a_1 = 6$	
2-й шаг цикла	$i=1+1=2$	$2 \leq 10$ ДА			$a_2 = -8$	
3-й шаг цикла	$i=2+1=3$	$3 \leq 10$ ДА			$a_3 = 15$	
4-й шаг цикла	$i=3+1=4$	$4 \leq 10$ ДА			$a_4 = 9$	
5-й шаг цикла	$i=4+1=5$	$5 \leq 10$ ДА			$a_5 = -1$	
6-й шаг цикла	$i=5+1=6$	$6 \leq 10$ ДА			$a_6 = 0$	
7-й шаг цикла	$i=6+1=7$	$7 \leq 10$ ДА			$a_7 = 5$	
8-й шаг цикла	$i=7+1=8$	$8 \leq 10$ ДА			$a_8 = -10$	
9-й шаг цикла	$i=8+1=9$	$9 \leq 10$ ДА			$a_9 = 12$	
10-й шаг цикла	$i=9+1=10$	$10 \leq 10$ ДА			$a_{10} = 2$	
Выход из цикла	$i=10+1=11$	$11 \leq 10$ НЕТ				
До цикла						$L=1$
1-й шаг цикла	$i=1$	$1 \leq 10$ ДА	$a_1=6 < 0$ НЕТ			
2-й шаг цикла	$i=1+1=2$	$2 \leq 10$ ДА	$a_2=-8 < 0$ ДА	$d=a_i=a_2=-8$	В цикле $j=i, L+1, -1$ $j=2, 2, -1$ Сдвиг $a_j=a_{j-1}$ $a_2=a_1=6$ затем $a_L=d$ $a_1=-8$	$L=2$
3-й шаг цикла	$i=2+1=3$	$3 \leq 10$ ДА	$a_3=15 < 0$ НЕТ			
4-й шаг цикла	$i=3+1=4$	$4 \leq 10$ ДА	$a_4=9 < 0$ НЕТ			
5-й шаг цикла	$i=4+1=5$	$5 \leq 10$ ДА	$a_5=-1 < 0$ ДА	$d=a_i=a_5=-1$	В цикле $j=i, L+1, -1$ $j=5, 3, -1$ Сдвиг $a_j=a_{j-1}$ $a_5=a_4=9$ $a_4=a_3=15$ $a_3=a_2=6$ затем $a_L=d$ $a_2=-1$	$L=3$
6-й шаг цикла	$i=5+1=6$	$6 \leq 10$ ДА	$a_6=0 < 0$ НЕТ			
7-й шаг цикла	$i=6+1=7$	$7 \leq 10$ ДА	$a_7=5 < 0$ НЕТ			

Продолжение табл. 1.19

Шаг алгоритма	i	Проверка условия выхода из цикла $i \leq n$	Проверка числа $a_i < 0$	d	a_i	L
8-й шаг цикла	$i=7+1=8$	$8 \leq 10$ ДА	$a_8=-10 < 0$ ДА	$d=a_i=a_8=-10$	В цикле $j=i, L+1, -1$ $j=8, 4, -1$ Сдвиг $a_j=a_{j-1}$ $a_8=a_7=5$ $a_7=a_6=0$ $a_6=a_5=9$ $a_5=a_4=15$ $a_4=a_3=6$ затем $a_L=d$ $a_3=-10$	$L=4$
9-й шаг цикла	$i=8+1=9$	$9 \leq 10$ ДА	$a_9=12 < 0$ НЕТ			
10-й шаг цикла	$i=9+1=10$	$10 \leq 10$ ДА	$a_{10}=2 < 0$ НЕТ			
Выход из цикла	$i=10+1=11$	$11 \leq 10$ НЕТ				
Результат	$a_1=-8, a_2=-1, a_3=-10, a_4=6, a_5=15, a_6=9, a_7=0, a_8=5, a_9=12, a_{10}=2$					

1.7.3. АЛГОРИТМЫ СОРТИРОВКИ ОДНОМЕРНЫХ МАССИВОВ

Сортировка элементов массива

Смысл любой сортировки заключается в перестановке элементов массива в определенном заданном порядке. Цель сортировки – облегчить последующий поиск элементов в таком отсортированном массиве.

Дано (входные данные): a_i , где $i=1, 2, \dots, n$.

Найти (выходные данные): a_i , где $i=1, 2, \dots, n$, такие, что

$a_1 \geq a_2 \geq a_3 \geq \dots \geq a_n$, если сортировка по убыванию;

$a_1 \leq a_2 \leq a_3 \leq \dots \leq a_n$, если сортировка по возрастанию.

Упорядочение осуществляется в процессе многократного просмотра исходного массива. Алгоритм должен быть таким, чтобы сортировка не требовала дополнительного массива, то есть проходила в исходном массиве.

Имеется большое разнообразие методов сортировки, каждый хорош в своем конкретном случае: для определенных данных, сортируемых на определенном компьютере с определенной целью. Нет универсального алгоритма сортировки наилучшего в любой ситуации. Рассмотрим для примера два метода сортировки.

Простой выбор

Идея сортировки, например, по возрастанию, состоит в следующем (рис. 1.47):

- в последовательности $a_1, a_2, a_3, a_4, a_5, \dots, a_{n-1}, a_n$ находится элемент с наименьшим значением и меняется местами с первым элементом a_1 ;
- далее те же действия выполняются с остальными $n-1$ элементами, то есть в последовательности $a_2, a_3, a_4, a_5, \dots, a_{n-1}, a_n$ находится элемент с наименьшим значением и меняется местами уже со вторым элементом a_2 ;
- затем в последовательности $a_3, a_4, a_5, \dots, a_{n-1}, a_n$ находится элемент с наименьшим значением и меняется местами с третьим элементом a_3 ;
- и так далее, пока не останется один элемент, наибольший.

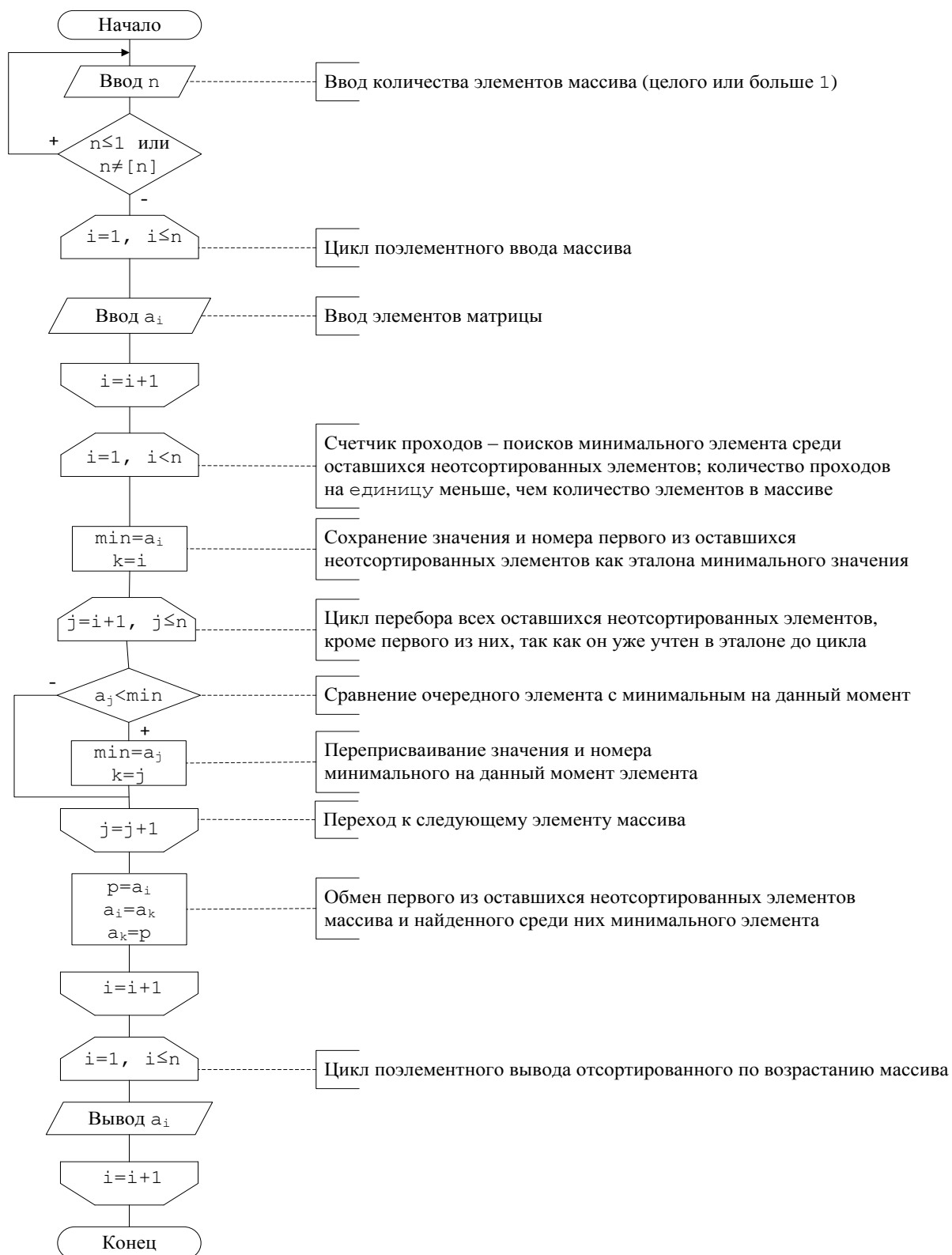


Рис. 1.47. Сортировка по возрастанию простым выбором

Если при поиске наименьшего элемента в части массива, окажется, что есть несколько элементов с одинаковым минимальным значением, то необходимо взять первый из них. Таким образом, можно достичь устойчивости сортировки. Метод сортировки является устойчивым, если относительный порядок элементов с равными значениями не меняется после упорядочивания.

Этот алгоритм достаточно простой для реализации. Работает достаточно быстро, если последовательность частично упорядочена, а также при малых значениях n . В среднем выполняется $n \cdot (n-1) / 2$ сравнений и $3(n-1)$ присваиваний.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив:

Индекс элемента	1	2	3	4	5	6	7
Значение элемента	4	17	25	1	21	14	2

Отсортированный массив:

Индекс элемента	1	2	3	4	5	6	7
Значение элемента	1	2	4	14	17	21	25

Выполним «ручную отладку» алгоритма для массива из тестового примера (табл. 1.20).

Таблица 1.20

Сортировка элементов массива простым выбором

Шаг алгоритма	Значения элементов со следующими индексами						
	1	2	3	4	5	6	7
До сортировки	4	17	25	1	21	14	2
i=1	j от 2 до 7			min			
После 1-го просмотра и обмена	1	17	25	4	21	14	2
i=2		j от 3 до 7					min
После 2-го просмотра и обмена	1	2	25	4	21	14	17
i=3			j от 4 до 7	min			
После 3-го просмотра и обмена	1	2	4	25	21	14	17
i=4				j от 5 до 7		min	
После 4-го просмотра и обмена	1	2	4	14	21	25	17
i=5					j от 6 до 7		min
После 5-го просмотра и обмена	1	2	4	14	17	25	21
i=6						j от 7 до 7	min
Результат сортировки	1	2	4	14	17	21	25

Простой обмен (пузырьковая сортировка)

Идея сортировки, например, по возрастанию, состоит в следующем (рис. 1.48):

- просматриваются элементы последовательности от начала к концу (можно и от конца к началу) и поочередно сравниваются соседние элементы a_1 и a_2 , a_2 и a_3 , a_3 и a_4 , a_4 и a_5 , ..., a_{n-1} и a_n ;
- если два соседних элемента расположены не по порядку, то они меняются местами. То есть если $a_{i-1} > a_i$, то они меняются местами;
- в результате такого одного прохода, максимальный элемент переместится на последнее место a_n ;

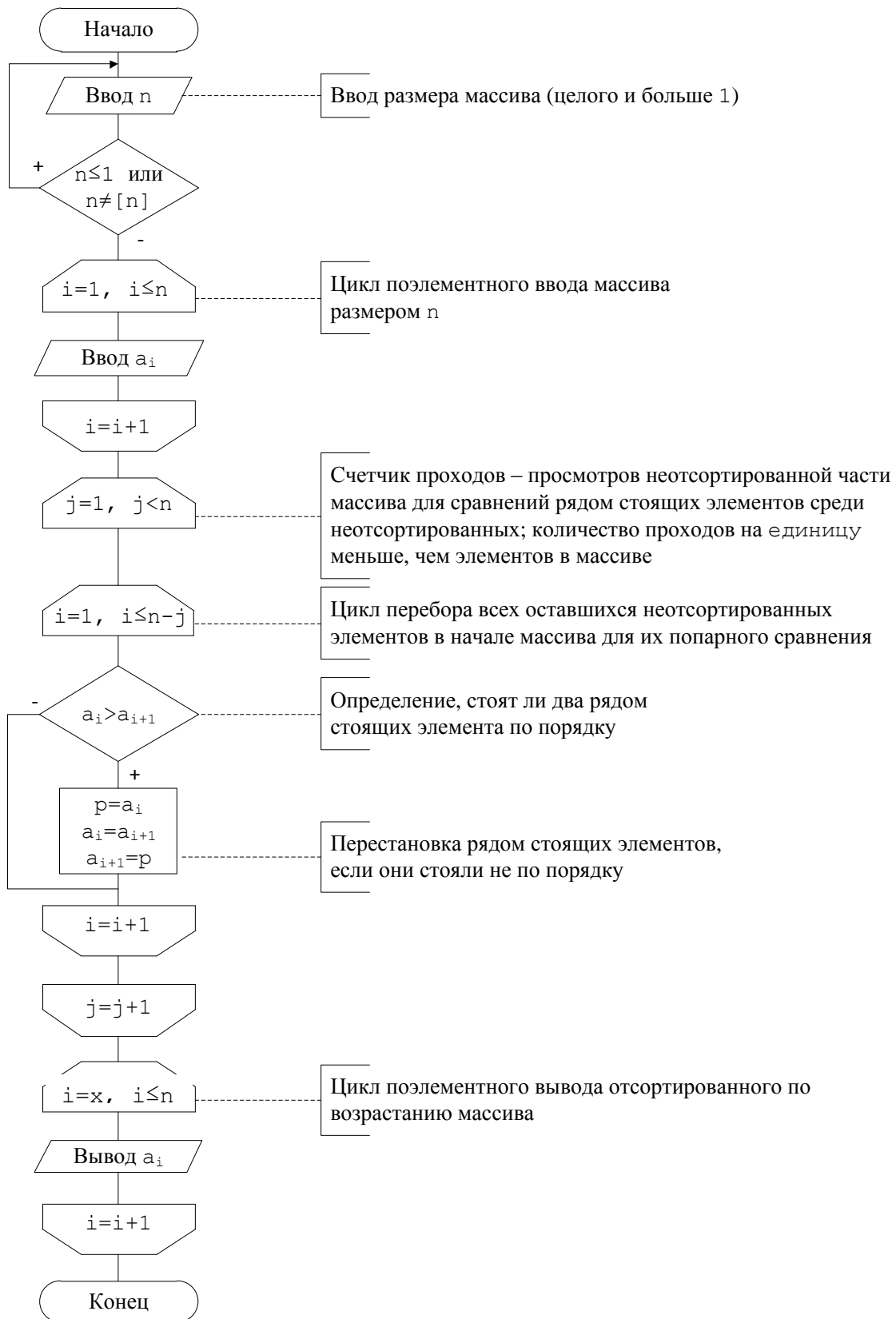


Рис. 1.48. Пузырьковая сортировка по возрастанию

• следующий проход (просмотр элементов попарно) уже не будет включать последний элемент, то есть будут поочередно сравниваться соседние элементы a_1 и a_2 , a_2 и a_3 , a_3 и a_4 , a_4 и a_5 , ..., a_{n-2} и a_{n-1} . И будет определен a_{n-1} – самый большой в оставшейся части последовательности;

• этот процесс повторяется $n-1$ раз до тех пор, пока все элементы не будут упорядочены.

Алгоритм осуществляет максимально возможное количество сравнений. Много раз приходится просматривать последовательность и выполнять много перестановок, что дает повод считать его неэффективным.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив:

Индекс элемента	1	2	3	4	5	6	7
Значение элемента	4	17	25	1	21	14	2

Отсортированный массив:

Индекс элемента	1	2	3	4	5	6	7
Значение элемента	1	2	4	14	17	21	25

Выполним «ручную отладку» алгоритма для массива из тестового примера (табл. 1.21).

Таблица 1.21

Пузырьковая сортировка элементов массива

Шаг алгоритма	Значения элементов со следующими индексами						
	1	2	3	4	5	6	7
До сортировки	4	17	25	1	21	14	2
1-й проход i от 1 до 6							
	4	17	25	1	21	14	2
$i=1$	Остаются						
$i=2$		Остаются					
$i=3$			Обмен				
	4	17	1	25	21	14	2
$i=4$				Обмен			
	4	17	1	21	25	14	2
$i=5$					Обмен		
	4	17	1	21	14	25	2
$i=6$						Обмен	
	4	17	1	21	14	2	25
2-й проход i от 1 до 5							
	4	17	1	21	14	2	25
$i=1$	Остаются						
$i=2$		Обмен					
	4	1	17	21	14	2	25
$i=3$			Остаются				
$i=4$				Обмен			
	4	1	17	14	21	2	25
$i=5$					Обмен		
	4	1	17	14	2	21	25

Продолжение табл. 1.21

Шаг алгоритма	Значения элементов со следующими индексами						
	1	2	3	4	5	6	7
3-й проход i от 1 до 4							
	4	1	17	14	2	21	25
	Обмен						
	1	4	17	14	2	21	25
	Остаются						
	Обмен						
	1	4	14	17	2	21	25
$i=4$	Обмен						
	1	4	14	2	17	21	25
4-й проход i от 1 до 3							
	1	4	14	2	17	21	25
$i=1$	Остаются						
$i=2$	Остаются						
$i=3$	Обмен						
	1	4	2	14	17	21	25
5-й проход i от 1 до 2							
	1	4	2	14	17	21	25
$i=1$	Остаются						
$i=2$	Обмен						
	1	2	4	14	17	21	25
6-й проход i от 1 до 1							
	1	2	4	14	17	21	25
$i=1$	Остаются						
Результат сортировки	1	2	4	14	17	21	25

1.7.4. АЛГОРИТМЫ РАБОТЫ С ДВУМЕРНЫМИ МАССИВАМИ

Рассмотрим несколько наиболее типичных алгоритмов работы с двумерными массивами.

Условимся во всех примерах *счет индексов вести с единицы*.

Перечислим некоторые *свойства матриц* (рис. 1.49):

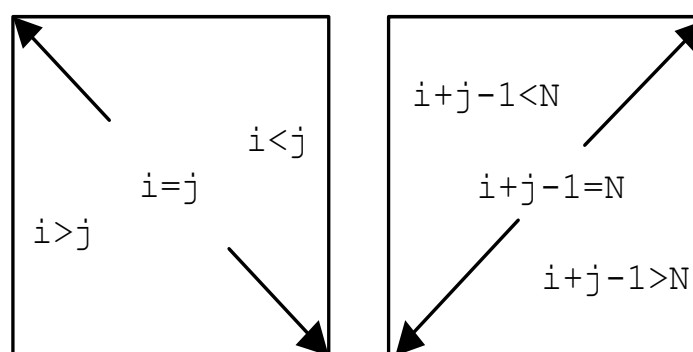


Рис. 1.49. Свойства элементов матрицы

- если номер строки элемента совпадает с номером столбца ($i=j$), это означает, что элемент лежит *на главной диагонали матрицы*;
- если номер строки превышает номер столбца ($i>j$), то элемент находится *ниже главной диагонали*;

- если номер столбца больше номера строки ($i < j$), то элемент находится **выше главной диагонали**.
- элемент лежит **на побочной диагонали**, если его индексы удовлетворяют равенству:

$$i + j - 1 = n;$$

- для элемента, находящегося **выше побочной диагонали**, характерно неравенство:

$$i + j - 1 < n;$$
- элементу, лежащему **ниже побочной диагонали**, соответствует выражение:

$$i + j - 1 > n.$$

Пример 27

Найти сумму элементов матрицы.

Дано (входные данные): a_{ij} , где $i=1, 2, \dots, n, j=1, 2, \dots, m$.

Найти (выходные данные): $S = \sum a_{ij}$, где $i=1, 2, \dots, n, j=1, 2, \dots, m$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив	Результат суммирования всех элементов массива
$\begin{pmatrix} 5 & 8 & 1 \\ 7 & 9 & 3 \end{pmatrix}$	$S = 5 + 8 + 1 + 7 + 9 + 3 = 33$

Алгоритм решения данной задачи (рис. 1.50) построен следующим образом:

- обнуляется ячейка для накапливания суммы (переменная S);
- затем с помощью двух вложенных циклов (внешний цикл по строкам – параметр цикла i , внутренний цикл по столбцам – параметр цикла j) просматривается каждый элемент матрицы a_{ij} и добавляется к сумме S .

Пример 28

Найти сумму элементов матрицы, расположенных выше главной диагонали (рис 1.51).

Дано (входные данные): a_{ij} , где $i=1, 2, \dots, n, j=1, 2, \dots, m$.

Найти (выходные данные): $S = \sum a_{ij}$, где $i=1, 2, \dots, n, j=1, 2, \dots, m, i < j$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм.

Исходный массив	Результат суммирования элементов массива, расположенных выше главной диагонали
$\begin{pmatrix} 7 & 4 & 2 \\ 8 & 9 & 5 \\ 1 & 3 & 6 \end{pmatrix}$	$S = 4 + 2 + 5 = 11$

Алгоритм решения данной задачи (рис. 1.52) построен следующим образом:

- обнуляется ячейка для накапливания суммы (переменная S);
- затем с помощью двух вложенных циклов (внешний по строкам, внутренний по столбцам) просматривается каждый элемент матрицы a_{ij} , но суммирование происходит только в том случае если, этот элемент находится выше главной диагонали, то есть выполняется свойство $i < j$.

На рисунке 1.53 изображен еще один вариант решения данной задачи.

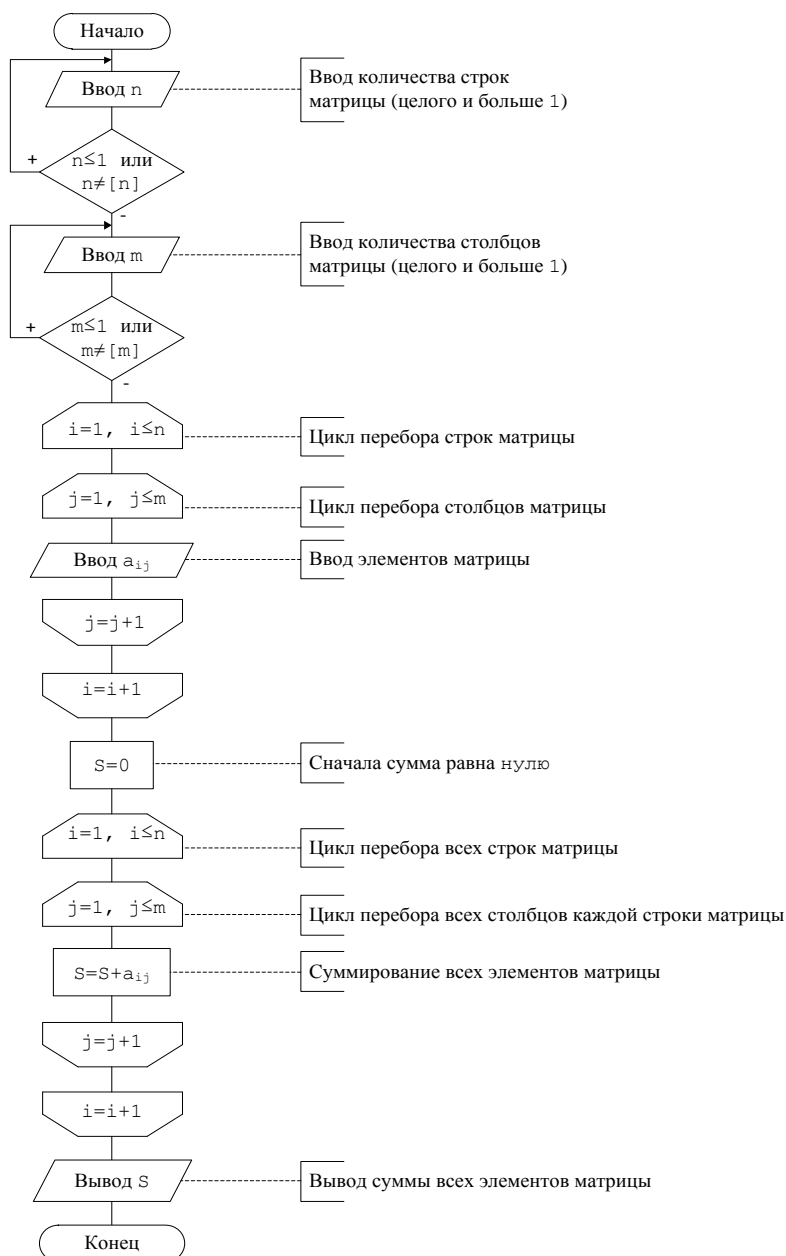


Рис. 1.50. Нахождение суммы элементов матрицы

В нем проверка условия $i < j$ не выполняется, но, тем не менее, в нем также суммируются элементы матрицы, находящиеся выше главной диагонали.

Для того чтобы понять, как работает этот алгоритм, внимательно рассмотрим рисунок 1.51. В первой строке заданной матрицы необходимо сложить все элементы, начиная со второго. Во второй строке — все элементы, начиная с третьего, в i -й строке процесс начнется с $(i+1)$ -го элемента, и так далее.

Таким образом, внешний цикл работает от 1 до n , а внутренний от $i+1$ до m .

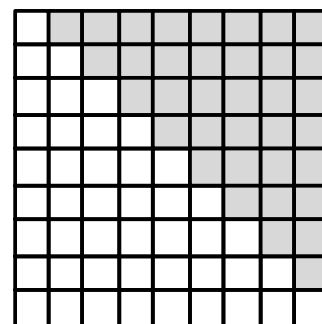
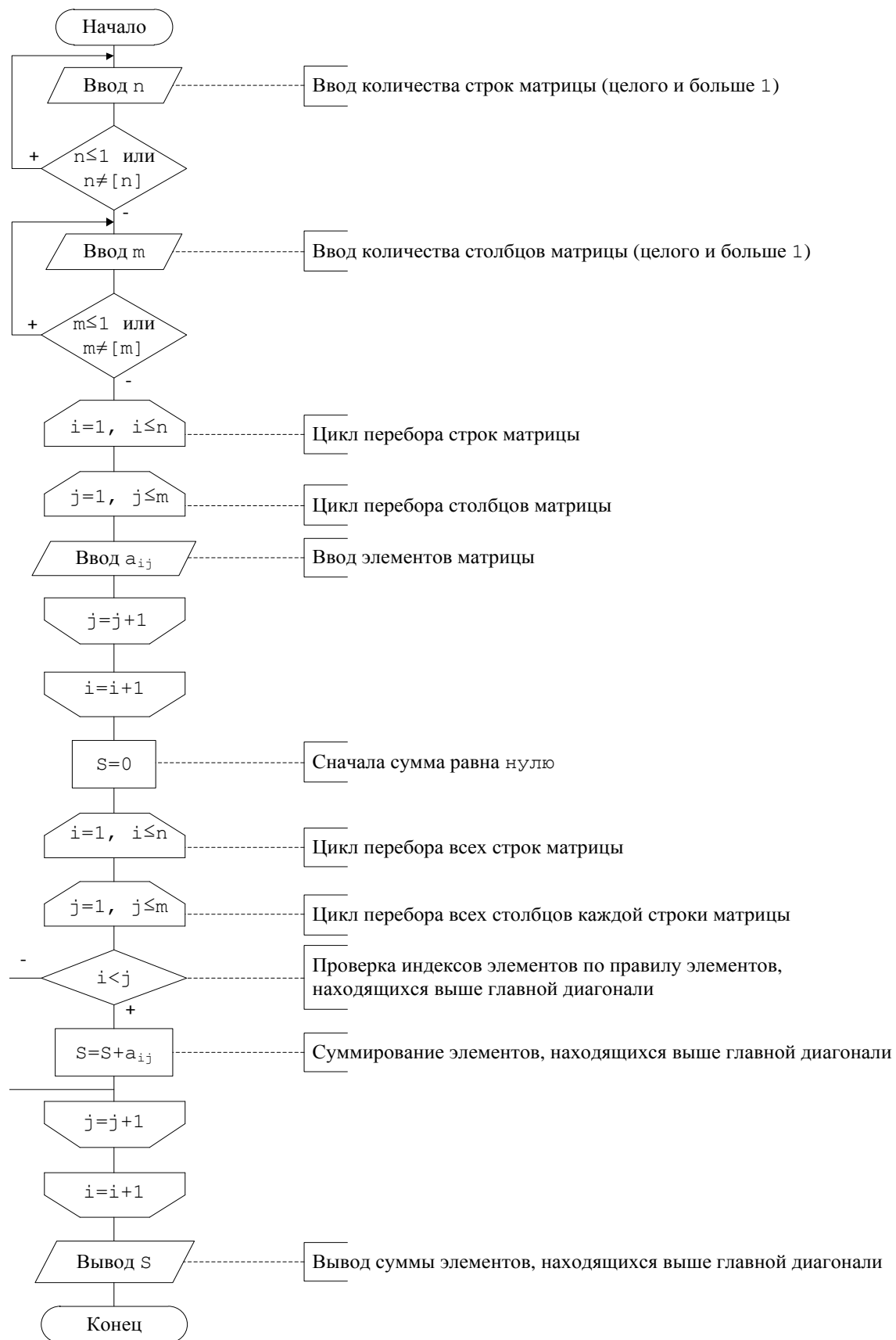


Рис. 1.51. Элементы матрицы, расположенные выше главной диагонали



Рис

. 1.52. «Неудачный» алгоритм нахождения суммы элементов, расположенных выше главной диагонали матрицы

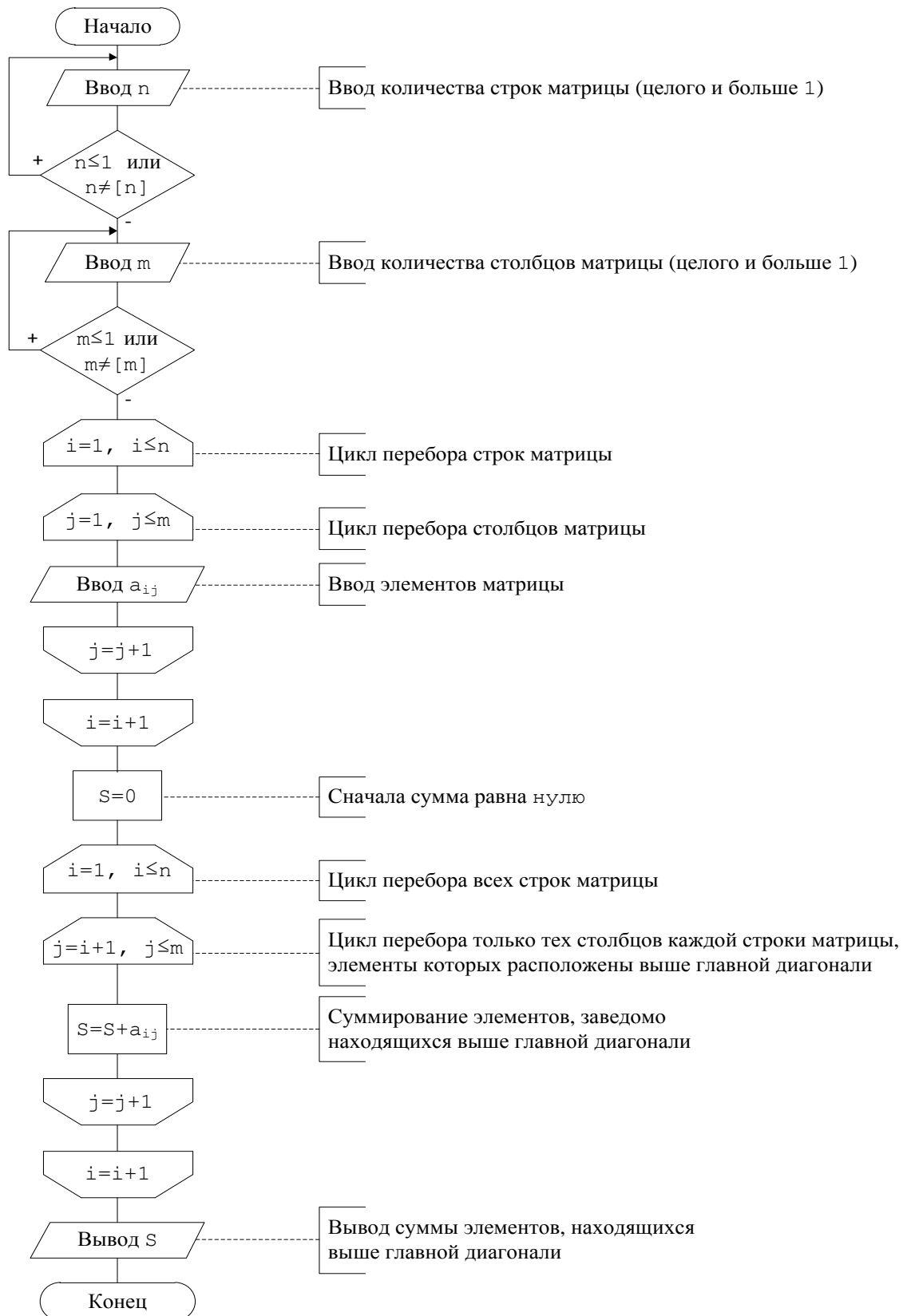


Рис. 1.53. «Удачный» алгоритм нахождения суммы элементов, расположенных выше главной диагонали матрицы

Пример 29

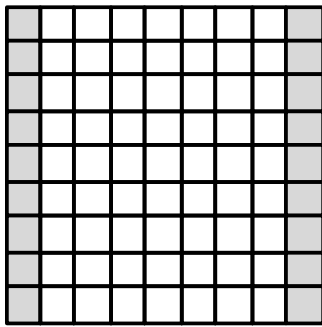


Рис. 1.54. Элементы матрицы, расположенные в первом и последнем столбце

Поменять местами 1-й и n -й столбцы матрицы a .

Дано (входные данные): a_{ij} , где $i=1, 2, \dots, k$, $j=1, 2, \dots, n$.

Найти (выходные данные): a_{ij} , где $i=1, 2, \dots, k$, $j=1, 2, \dots, n$.

Прежде, чем составлять алгоритм рассмотрим рисунок 1.54, на котором изображена схема матрицы с выделенными первым и последним столбцами.

Составим также тестовый пример, чтобы более наглядно представить себе алгоритм:

Исходный массив	Результат перестановки первого и последнего столбцов
$\begin{pmatrix} 1 & 2 & 0 & 8 \\ -3 & 5 & 7 & 6 \\ 2 & 3 & 9 & -1 \\ 7 & -2 & 2 & 4 \end{pmatrix}$	$\begin{pmatrix} 8 & 2 & 0 & 1 \\ 6 & 5 & 7 & -3 \\ -1 & 3 & 9 & 2 \\ 4 & -2 & 2 & 7 \end{pmatrix}$

Для решения данной задачи не принципиально, квадратная матрица задана или нет. Блок-схема обмена столбцов матрицы приведена на рисунке 1.55.

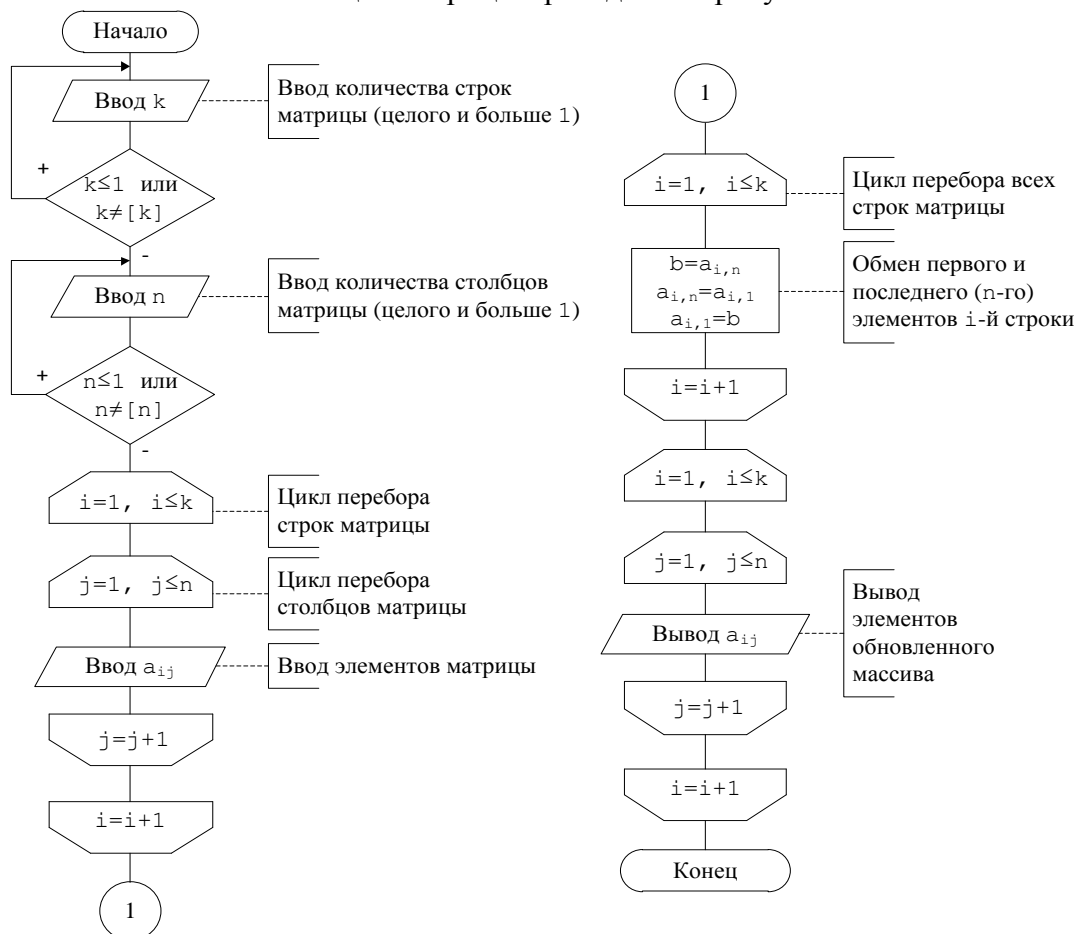


Рис. 1.55. Обмен первого и последнего столбцов матрицы

Пример 30.

Вычислить сумму и количество положительных элементов квадратной матрицы, расположенных по ее периметру и на диагоналях.

Дано (входные данные): a_{ij} , где $i=1, 2, \dots, n, j=1, 2, \dots, n$.

Найти (выходные данные): K – количество, $S = \sum a_{ij}$, где $i=1, 2, \dots, n; j=1, 2, \dots, n, a_{ij} > 0$ и расположены по периметру и на диагоналях матрицы.

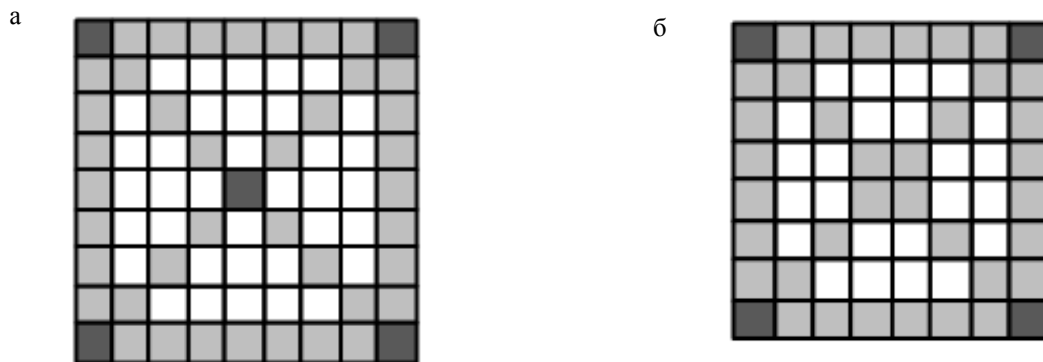


Рис. 1.56. Элементы квадратной матрицы, расположенные по периметру и на диагоналях: а - n – нечетное; б - n – четное

В квадратной матрице число строк равно числу столбцов.

Прежде чем приступить к решению задачи, рассмотрим схемы квадратных матриц различных размеров (рис. 1.56).

Из условия задачи понятно, что не нужно рассматривать все элементы заданной матрицы. Достаточно просмотреть первую и последнюю строки, первый и последний столбцы, а также диагонали. Все эти элементы отмечены на схеме, причем черным цветом выделены элементы, обращение к которым может произойти дважды.

Например, элемент с номером (1,1) принадлежит как к первой строке, так и к первому столбцу, а элемент с номером (n,n) находится в последней строке и последнем столбце одновременно. Кроме того, если n – число нечетное (см. рис. 1.56, а), то существует элемент с номером ($[n/2]+1, [n/2]+1$), который находится на пересечении главной и побочной диагоналей. При четном значении n (см. рис. 1.56, б) диагонали не пересекаются.

Составим тестовые примеры, чтобы более наглядно представить себе алгоритм:

1)

Исходный массив нечетного размера	Результат суммирования положительных элементов массива, расположенных по периметру и на диагоналях	
$\begin{pmatrix} 1 & -2 & 3 & 4 & -5 \\ 6 & 7 & 8 & -9 & 10 \\ -11 & -12 & 13 & 14 & 15 \\ 16 & 17 & -18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$	$S=1+7+13+19+25+$ $13+17+21+$ $3+4+$ $22+23+24+$ $6+16+$ $10+15+20$ -13 $=256$	Главная диагональ Побочная диагональ Первая строка Последняя строка Первый столбец Последний столбец Дважды учтенный a_{33} Результат

2)

Исходный массив четного размера	Результат суммирования положительных элементов массива, расположенных по периметру и на диагоналях	
$\begin{pmatrix} 1 & -2 & 3 & 4 & -5 & 6 \\ 7 & 8 & 9 & 10 & -11 & -12 \\ 13 & 14 & -15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \\ -25 & 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & -34 & 35 & 36 \end{pmatrix}$	$S=1+8+22+29+36+6+16+21+26+31+3+4+32+33+35+7+13+19+18+24+30=414$	Главная диагональ Побочная диагональ Первая строка Последняя строка Первый столбец Последний столбец Результат

Итак, разобрав подробно постановку задачи, составив тестовые примеры, рассмотрим алгоритм ее решения.

Для обращения к элементам главной диагонали вспомним, что номера строк этих элементов всегда равны номерам столбцов. Поэтому, если параметр i изменяется циклически от 1 до n , то $a_{i,i}$ – элемент главной диагонали.

Воспользовавшись свойством, характерным для элементов побочной диагонали, получим:

$$i+j-1=n,$$

отсюда

$$j=n-i+1.$$

Следовательно, для $i=1, 2, \dots, n$ элемент $a_{i,n-i+1}$ – элемент побочной диагонали.

Элементы, находящиеся по периметру матрицы записываются следующим образом:

$a_{1,i}$ – первая строка; $a_{i,1}$ – первый столбец;
 $a_{n,i}$ – последняя строка; $a_{i,n}$ – последний столбец.

Блок-схема описанного алгоритма изображена на рисунке 1.57.

После ввода элементов матрицы первым организуется цикл для обращения к диагональным элементам матрицы. В этом цикле отдельно подсчитываются сумма и количество положительных элементов на главной диагонали и на побочной диагонали.

В следующем цикле изменение параметра i задается от 2 до $n-1$. Это необходимо для того, чтобы не обращаться к элементам, которые уже были рассмотрены ($a_{1,1}$, $a_{1,n}$, $a_{n,1}$ и $a_{n,n}$). В этом цикле отдельно подсчитываются сумма и количество положительных элементов в первой строке, в последней строке, в первом столбце, в последнем столбце.

После этих циклов стоит проверка, не был ли элемент, находящийся на пересечении диагоналей, учтен дважды. Напомним, что это могло произойти только в том случае, если n – нечетное число и этот элемент был положительным. Эти условия и проверяются. В том случае, если это ИСТИНА, то происходит уменьшение вычисленного количества положительных элементов на единицу и уменьшение суммы на значение дважды учтенного элемента матрицы.

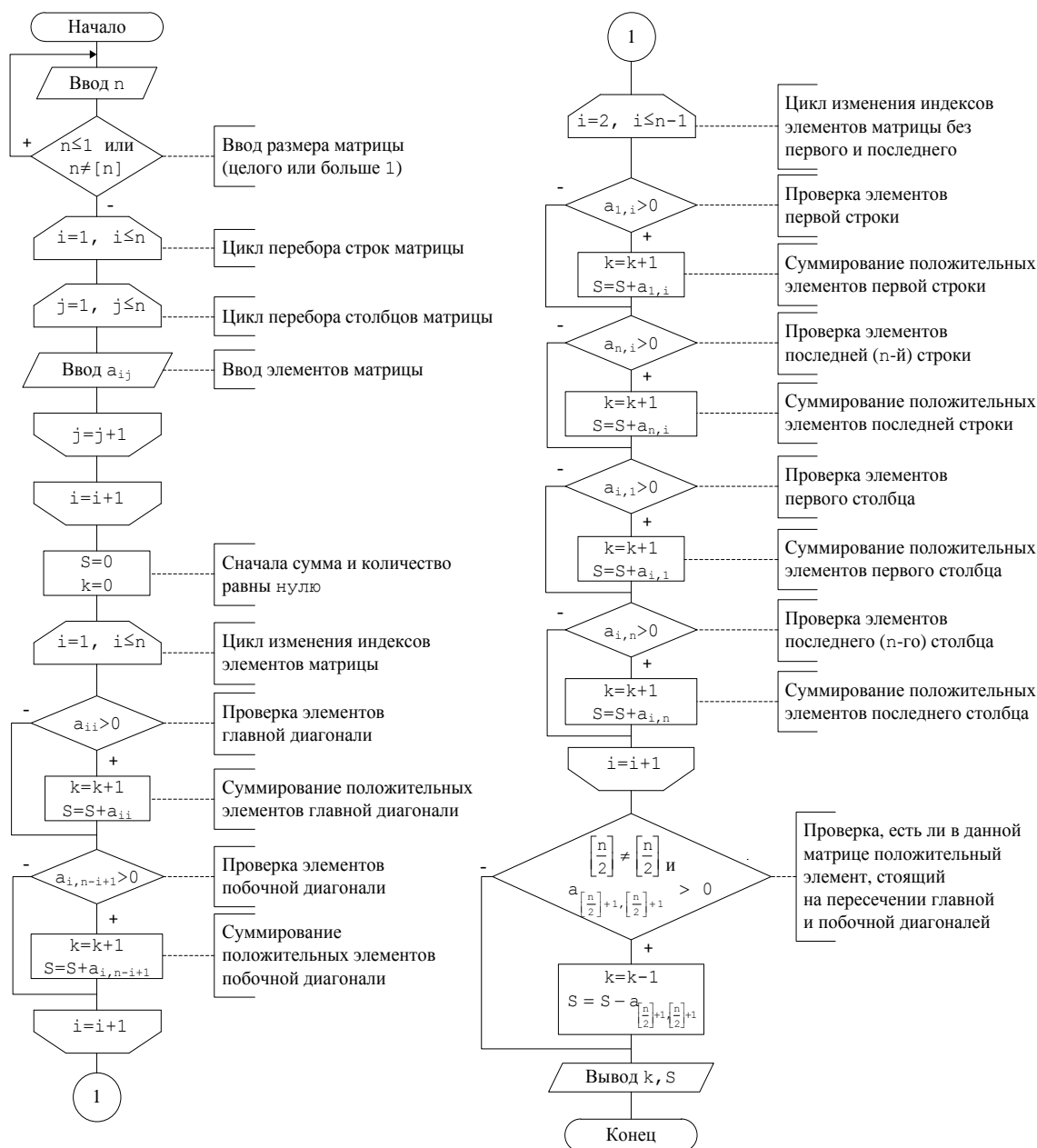


Рис. 1.57. Определение суммы и количества положительных элементов матрицы, расположенных по ее периметру и на диагоналях

Пример 31

Задана матрица **a**. Сформировать вектор **p**, в который *записать номера строк максимальных элементов* каждого столбца.

Дано (входные данные): a_{ij} , где $i=1, 2, \dots, n, j=1, 2, \dots, m$.

Найти (выходные данные): $p_j=i$, где i – индекс элемента $a_{ij}=\max\{a_{1j}, a_{2j}, \dots, a_{nj}\}$, для $j=1, 2, \dots, m$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм:

Исходный массив	Номер столбца массива	Максимальный элемент столбца			Результи- рующий вектор- столбец
		Значение	Обозначение	Номер строки	
$\begin{pmatrix} 1 & 5 & 0 & 8 \\ -3 & 2 & 9 & 6 \\ 2 & -2 & 7 & -1 \\ 7 & 3 & 2 & 4 \end{pmatrix}$	1	7	a_{41}	4	$\begin{pmatrix} 4 \\ 1 \\ 2 \\ 1 \end{pmatrix}$
	2	5	a_{12}	1	
	3	9	a_{23}	2	
	4	8	a_{14}	1	

Алгоритм решения этой задачи следующий: для каждого столбца матрицы находим максимальный элемент и его номер, номер максимального элемента j -го столбца матрицы записываем в j -й элемент массива p . Блок-схема алгоритма приведена на рисунке 1.58.

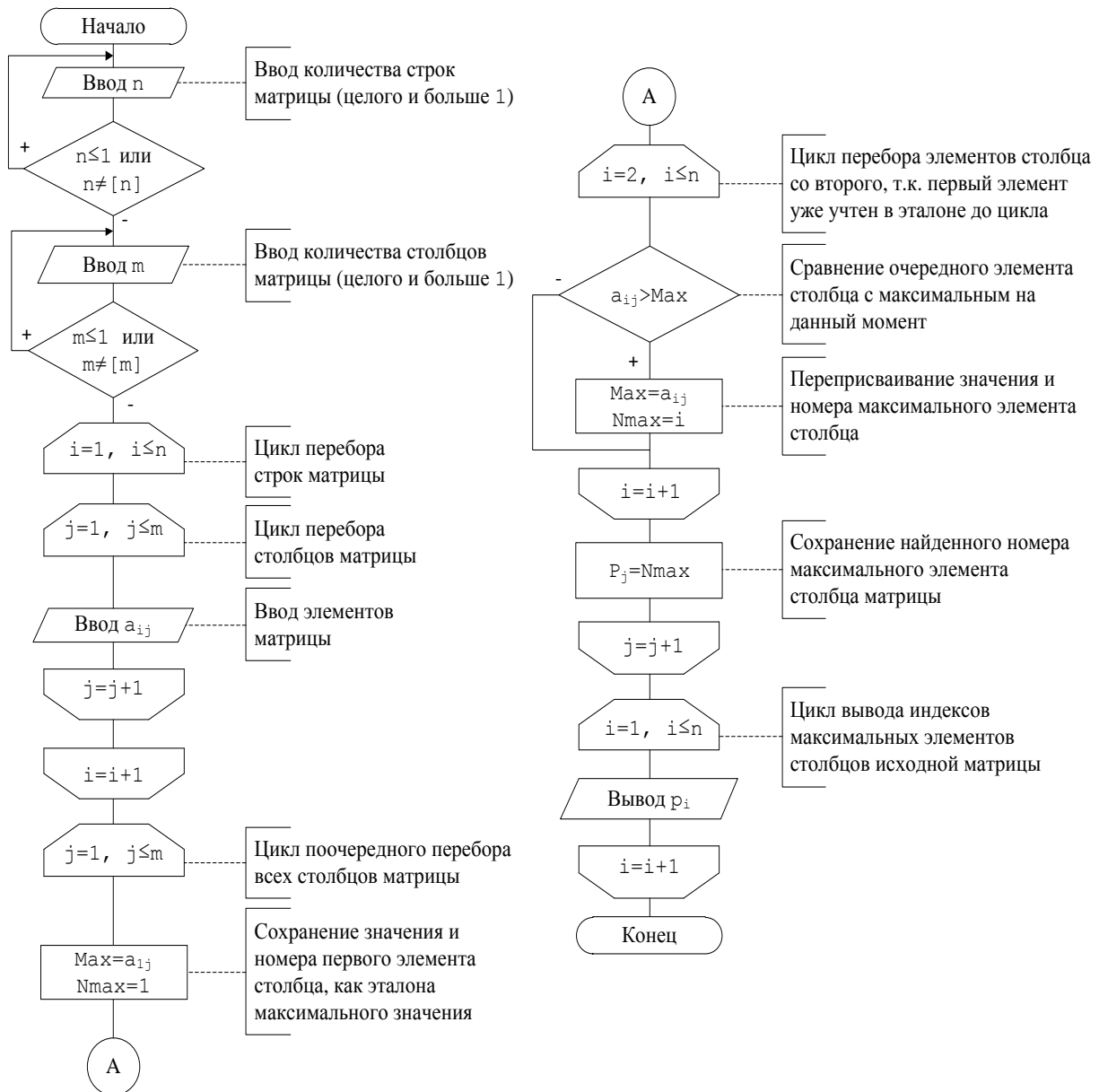


Рис. 1.58. Поиск номеров строк максимальных элементов каждого столбца матрицы

Пример 32

Преобразовать матрицу **a** таким образом, чтобы *каждый столбец был упорядочен по убыванию*.

Дано (входные данные): a_{ij} , где $i=1, 2, \dots, n, j=1, 2, \dots, m$.

Найти (выходные данные): a_{ij} , такие, что $a_{1j} \geq a_{2j} \geq a_{3j} \geq \dots \geq a_{nj}$, для $j=1, 2, \dots, m$, где $i=1, 2, \dots, n$.

Составим тестовый пример, чтобы более наглядно представить себе алгоритм:

Исходный массив	Результат сортировки каждого столбца по убыванию
$\begin{pmatrix} 1 & 5 & 0 & 8 \\ -3 & 2 & 9 & 6 \\ 2 & -2 & 7 & -1 \\ 7 & 3 & 2 & 4 \end{pmatrix}$	$\begin{pmatrix} 7 & 5 & 9 & 8 \\ 2 & 3 & 7 & 6 \\ 1 & 2 & 2 & 4 \\ -3 & -2 & 0 & -1 \end{pmatrix}$

Алгоритм решения этой задачи сводится к тому, что уже известный алгоритм упорядочивания элементов в массиве выполняется для каждого столбца матрицы. Блок-схема приведена на рисунке 1.59. Используется метод сортировки пузырьком.

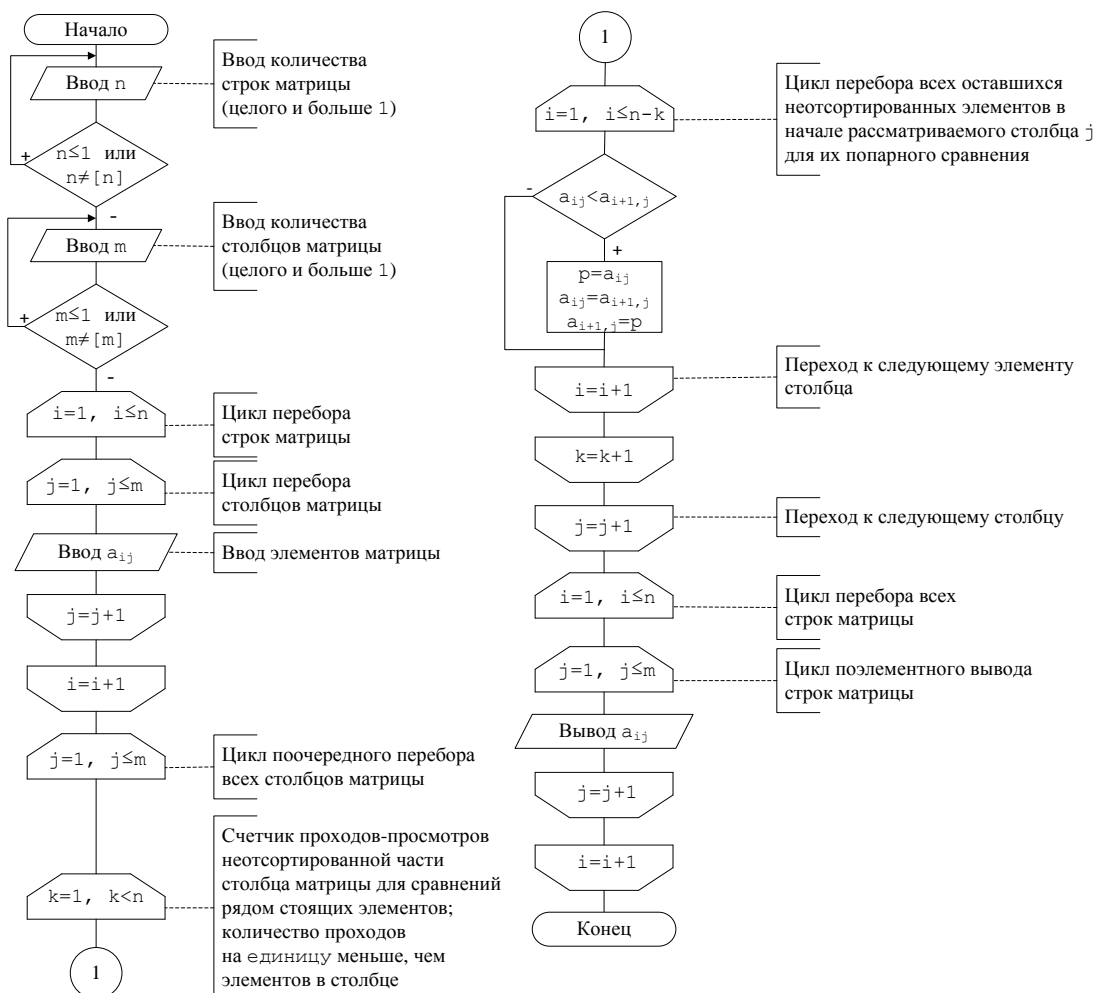


Рис. 1.59. Сортировка столбцов матрицы по убыванию

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ

Расчетно-графическое задание *оформляется в тетради в клетку с полями*.

Расчетно-графическое задание выполняется письменно от руки разборчивым почерком, чернилами любого цвета (кроме красного).

Обложка оформляется по единым правилам оформления расчетно-графических заданий РД ФГБОУ ВПО «КНАГТУ» 013-2011 «Текстовые студенческие работы. Правила оформления».

На обложке необходимо указать дату выполнения работы и поставить свою подпись.

Задания, включенные в РГЗ, выполняются строго в заданной последовательности. Номера заданий следует сохранить. Каждое задание в работе записывается с новой страницы. Обязательно приводится текст задания (можно распечатать и вклеить). Если задание имеет общее условие, то при записывании условия задания общие данные заменяются конкретными из вашего варианта.

После получения проверенной работы вы должны исправить в ней помеченные рецензентом ошибки и недочеты.

Работа над ошибками, как правило, делается в той же тетради. При необходимости работу можно сделать в новой тетради, но при возвращении на рецензирование следует приложить первоначально рецензированную работу.

ЗАДАНИЕ 1

Выбрать правильный ответ из списка предложенных ответов.

В отчете привести: номер правильного ответа и обоснование Вашего выбора в свободной форме.

Внимание! Прежде, чем приступать к выполнению данного задания, решите как минимум примеры 2, 3, 4.

ВАРИАНТЫ ЗАДАНИЯ 1

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
01 15	Запись выражения $y = Ax^2 + Bx + C$ на алгоритмическом языке имеет вид... Варианты ответов: 1) $y = (A * x) ^ 2 + B * x + C$ 2) $y = Ax^2 + Bx + C$ 3) $y = Ax^2 + Bx + C$ 4) $y = A * x^2 + B * x + C$	03 12	Для определения сдачи с N рублей при покупке максимального числа единиц товара стоимостью k рублей за единицу может использоваться некоторая формула. Выберите номер правильного варианта формулы : 1) $N - \text{INT}(N/k) * k$ 2) $N - \text{INT}(k/N) * k$ 3) $N - \text{INT}(N/k) * N$ 4) $\text{INT}(N/k) * k$
02 11	В результате работы алгоритма $Y = X + 5$ $X = Y$ $Y = X + Y$ вывод Y переменная Y приняла значение 14 . Укажите значение переменной X до начала работы алгоритма. Варианты ответов:	04 13	Задан массив $A[1..4]$, состоящий из строк $A = (\langle 2000 \rangle, \langle 836 \rangle, \langle 102 \rangle, \langle 21 \rangle)$. После сортировки по убыванию элементы массива будут расположены в следующем порядке: Варианты ответов: 1) $\langle 2000 \rangle, \langle 836 \rangle, \langle 102 \rangle, \langle 21 \rangle$ 2) $\langle 836 \rangle, \langle 21 \rangle, \langle 2000 \rangle, \langle 102 \rangle$

	1) 7 2) 5 3) 10 4) 2		3) «836», «2000», «21», «102» 4) «21», «102», «836», «2000»
05 14	В результате работы алгоритма $Y = X - 5$ $X = 2 * (Y + 1)$ $Y = X + Y$ <u>Вывод</u> Y переменная Y приняла значение 5. Укажите число, которое являлось значением переменной X до начала работы алгоритма. Варианты ответов: 1) 5 2) 2 3) 6 4) 7	08 17	Круглые скобки для определения порядка выполнения вычислений выражения $a^b * 2 + 3.456y$ правильно расставлены в выражении... Варианты ответов: 1) $((A^B) * 2 + 3.456 * y)$ 2) $((a^b) * 2) + (3.456 * y)$ 3) $(A^ (B * 2) + 3.456 * y)$ 4) $A^ (B * 2) + (3.456 * y)$
06 00	Правильная запись выражения $Y = A^{x+1}B + 2C$ на алгоритмическом языке имеет вид... Варианты ответов: 1) $Y = A^ ((X + 1) * B) + 2 * C$ 2) $Y = A^ (X + 1) B + 2C$ 3) $Y = A^ (X + 1) * B + 2 * C$ 4) $Y = A^ X + 1 * B + 2 * C$	09 18	Задан массив A[1..4], состоящий из строк A = («1000», «836», «102», «21»). После сортировки по возрастанию элементы массива будут расположены в следующем порядке: Варианты ответов: 1) «1000», «836», «102», «21» 2) «1000», «102», «21», «836» 3) «21», «102», «836», «1000» 4) «21», «102», «1000», «836»
07 16	В результате работы алгоритма $Y = X - 1$ $X = Y + 2$ $Y = X + Y$ <u>Вывод</u> Y переменная Y приняла значение 10. Укажите число, которое являлось значением переменной X до начала работы алгоритма. Варианты ответов: 1) 5 2) 7 3) 2 4) 10	10 19	В результате работы алгоритма $Y = X + 3$ $X = 2 * Y$ $Y = X + Y$ <u>Вывод</u> Y переменная Y приняла значение 18. Укажите число, которое являлось значением переменной X до начала работы алгоритма. Варианты ответов: 1) 10 2) 3 3) 5 4) 7

ЗАДАНИЕ 2

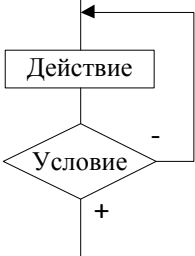
Выбрать правильный ответ из списка предложенных ответов.

В отчете привести: номер правильного ответа и обоснование Вашего выбора в свободной форме.

Внимание! Прежде, чем приступить к выполнению данного задания, прочитайте как минимум подразд. 1.6.2, 1.6.3 и решите как минимум примеры 5, 9, 10.

ВАРИАНТЫ ЗАДАНИЯ 2

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
01 14	<p>Следующий фрагмент алгоритма имеет _____ структуру.</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Циклическую с предусловием 2) Циклическую с постусловием 3) Линейную 4) Разветвляющуюся 	04 12	<p>На блок-схеме представлена базовая алгоритмическая конструкция _____.</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Цикл с повторением 2) Цикл с предусловием 3) Цикл с постусловием 4) Ветвление
02 15	<p>Следующий фрагмент алгоритма имеет _____ структуру.</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Циклическую с предусловием 2) Циклическую с постусловием 3) Линейную 4) Разветвляющуюся 	05 13	<p>Следующий фрагмент программы</p> <pre> ЕСЛИ X>Y ТО ЕСЛИ X>Z ТО M=X ИНАЧЕ M=Z ВСЕ ИНАЧЕ ЕСЛИ Y>Z ТО M=Y ИНАЧЕ M=Z ВСЕ ВСЕ </pre> <p>вычисляет...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) максимум из трех чисел 2) наибольшее из чисел X и Y 3) наименьшее из чисел Y и Z 4) минимум из трех чисел
03 11	<p>Если элементы массива D[1..5] равны соответственно 3, 4, 5, 1, 2, то значение выражения D[D[3]] - D[D[5]] равно...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) 3 2) -2 3) 2 4) -1 	06 19	<p>Для значения переменной X=14 результатом выполнения алгоритма, представленного фрагментом блок-схемы, будет величина, равная _____.</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) S=-1 2) S=14 3) S= 0 4) S= 1

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
07 00	<p>Следующий фрагмент алгоритма имеет _____ структуру.</p>  <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Циклическую с предусловием 2) Циклическую с постусловием 3) Линейную 4) Разветвляющуюся 	09 17	<p>После выполнения фрагмента алгоритма</p> <p>$x=2; a=20$</p> <p><u>Если ($x \leq 1$) и ($a \geq 19$)</u></p> <p><u>то</u> $x=2*a$</p> <p><u>иначе</u> $a=0; x=2*a$</p> <p><u>все</u></p> <p>значения переменных <i>x</i> и <i>a</i> стали...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) $x=0; a=0$ 2) $x=2; a=20$ 3) $x=40; a=20$ 4) $x=0; a=40$
08 16	<p>Максимальное значение из переменных <i>X</i> и <i>Y</i> определяет оператор...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Если $X > Y$, то $MAX=X$, иначе $MAX=Y$ 2) $MAX = (X-Y) / 2 + (X+Y) / 2$ 3) Если $X > Y$, то $MAX=X$ 4) Если $X < Y$, то $MAX=Y$ 	10 18	<p>Следующий фрагмент программы</p> <p>ЕСЛИ $X < Y$ ТО</p> <p> ЕСЛИ $X < Z$ ТО $M=X$</p> <p> ИНАЧЕ $M=Z$</p> <p> ВСЕ</p> <p>ИНАЧЕ</p> <p> ЕСЛИ $Y < Z$ ТО $M=Y$</p> <p> ИНАЧЕ $M=Z$</p> <p> ВСЕ</p> <p>ВСЕ</p> <p>вычисляет...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) максимум из трех чисел 2) минимум из трех чисел 3) наибольшее из чисел X и Y 4) наименьшее из чисел Y и Z

ЗАДАНИЕ 3

Выбрать правильный ответ из списка предложенных ответов.

В отчете привести: номер правильного ответа и обоснование вашего выбора в свободной форме.

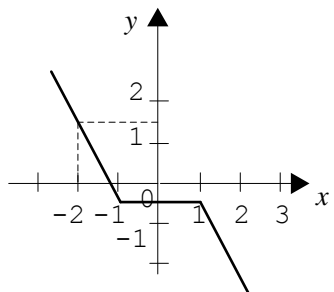
Примечание: Операция ***mod* (*x*, *y*)** – получение остатка целочисленного деления ***x*** на ***y***.

Операция ***div* (*x*, *y*)** – целочисленное деление ***x*** на ***y***.

Внимание! Прежде, чем приступить к выполнению данного задания, решите как минимум примеры 1, 7, 18.

ВАРИАНТЫ ЗАДАНИЯ 3

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
01 13	<p>После выполнения следующего фрагмента алгоритма k=70</p> <p>выбор</p> <p> <u>при</u> $\text{mod}(k, 12) = 7$ $d = k$; <u>при</u> $\text{mod}(k, 12) < 5$ $d = 2$; <u>при</u> $\text{mod}(k, 12) > 9$ $d = 3$; иначе $d = 1$;</p> <p>все</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <p>1) 3 2) 70 3) 2 4) 1</p>	04 11	<p>После выполнения следующего фрагмента алгоритма k=30</p> <p>выбор</p> <p> <u>при</u> $\text{div}(k, 12) = 2$ $d = k$; <u>при</u> $\text{mod}(k, 12) < 5$ $d = 2$; <u>при</u> $\text{mod}(k, 12) > 9$ $d = 3$; иначе $d = 1$;</p> <p>все</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <p>1) 2 2) 3 3) 30 4) 1</p>
02 14	<p>После выполнения следующего фрагмента алгоритма k=50</p> <p>выбор</p> <p> <u>при</u> $\text{mod}(k, 12) = 7$ $d = k$; <u>при</u> $\text{mod}(k, 12) < 5$ $d = 2$; <u>при</u> $\text{mod}(k, 12) > 9$ $d = 3$; иначе $d = 1$;</p> <p>все</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <p>1) 2 2) 50 3) 3 4) 1</p>	05 12	<p>После выполнения следующего фрагмента алгоритма k=40</p> <p>выбор</p> <p> <u>при</u> $\text{div}(k, 12) = 2$ $d = k$; <u>при</u> $\text{mod}(k, 12) < 5$ $d = 2$; <u>при</u> $\text{mod}(k, 12) > 9$ $d = 3$; иначе $d = 1$;</p> <p>все</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <p>1) 40 2) 1 3) 2 4) 3</p>
03 15	<p>После выполнения следующего фрагмента алгоритма k=30</p> <p>выбор</p> <p> <u>при</u> $\text{mod}(k, 12) = 7$ $d = k$; <u>при</u> $\text{mod}(k, 12) < 5$ $d = 2$; <u>при</u> $\text{mod}(k, 12) > 9$ $d = 3$; иначе $d = 1$;</p> <p>все</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <p>1) 30 2) 3 3) 2 4) 1</p>	06 18	<p>После выполнения следующего фрагмента алгоритма k=50</p> <p>выбор</p> <p> <u>при</u> $\text{div}(k, 12) = 4$ $d = k$; <u>при</u> $\text{div}(k, 12) < 5$ $d = 2$; <u>при</u> $\text{mod}(k, 12) > 9$ $d = 3$; иначе $d = 1$;</p> <p>все</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <p>1) 1 2) 2 3) 3 4) 50</p>

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
07 19	<p>После выполнения следующего фрагмента алгоритма $k=50$ выбор <u>при</u> $\text{div}(k, 12)=4$ $d=k$; <u>при</u> $\text{mod}(k, 12)<5$ $d=2$; <u>при</u> $\text{mod}(k, 12)>9$ $d=3$; иначе $d=1$; все значение переменной d равно...</p> <p>Варианты ответов: 1) 2 2) 3 3) 1 4) 50</p>	09 16	<p>Определите значения переменных p и d после выполнения следующего фрагмента алгоритма при условии $k=65$. выбор <u>при</u> $\text{mod}(k, 12)=7$ $d=k$: $p=\underline{\text{да}}$; <u>при</u> $\text{mod}(k, 12)<5$ $d=2$: $p=\underline{\text{нет}}$; <u>при</u> $\text{mod}(k, 12)>9$ $d=3$: $p=\underline{\text{нет}}$; иначе $d=1$: $p=\underline{\text{да}}$; все</p> <p>Варианты ответов: 1) $p=\text{нет}$; $d=2$ 2) $p=\text{да}$; $d=1$ 3) $p=\text{да}$; $d=65$ 4) $p=\text{нет}$; $d=3$</p>
08 00	<p>После выполнения следующего фрагмента алгоритма $k=30$ выбор <u>при</u> $\text{div}(k, 12)=4$ $d=k$; <u>при</u> $\text{div}(k, 12)<5$ $d=2$; <u>при</u> $\text{mod}(k, 12)>9$ $d=3$; иначе $d=1$; все значение переменной d равно...</p> <p>Варианты ответов: 1) 2 2) 50 3) 3 4) 1</p>	10 17	<p>Выберите вариант написания оператора, которым описывается следующий график.</p>  <p>Математическая модель: $Y = \begin{cases} A, & \text{если } x < -1; \\ B, & \text{если } x > 1; \\ C, & \text{в остальных случаях.} \end{cases}$</p> <p>Варианты ответов: 1) $A=-3.0 \cdot x-3.5$; $B=-x+0.5$; $C=-0.5$ 2) $A=-2.0 \cdot x-2.5$; $B=-x+0.5$; $C=-0.5$ 3) $A=-1.5 \cdot x-2$; $B=-0.5 \cdot x$; $C=-0.5$ 4) $A=-1.5 \cdot x-2$; $B=-2.0 \cdot x+1.5$; $C=-0.5$</p>

ЗАДАНИЕ 4

Выбрать правильный ответ из списка предложенных ответов.

В отчете привести: номер правильного ответа и обоснование Вашего выбора в свободной форме.

Внимание! Прежде, чем приступить к выполнению данного задания, решите как минимум примеры 9, 10, 11.

ВАРИАНТЫ ЗАДАНИЯ 4

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
01 12	После выполнения следующего алгоритма $b=11; d=46$ нц пока $d \geq b$ $ d=d-b$ кц значение переменной d равно... Варианты ответов: 1) 24 2) 2 3) 35 4) 13	04 15	В представленном фрагменте программы $b=10; d=50$ нц пока $d \geq b$ $ d=d-b$ кц тело цикла выполнится... Варианты ответов: 1) 1 раз 2) 4 раза 3) 5 раз 4) 2 раза
02 13	После выполнения следующего алгоритма $b=12; d=46$ нц пока $d \geq b$ $ d=d-b$ кц значение переменной d равно... Варианты ответов: 1) 34 2) 10 3) 22 4) 46	05 11	После выполнения алгоритма $b=10; d=30$ нц пока $d \geq b$ $ d=d-b$ кц значение переменной d равно... Варианты ответов: 1) 10 2) 30 3) 20 4) 0
03 14	После выполнения следующего алгоритма $b=10; d=50$ нц пока $d \geq b$ $ d=d-b$ кц значение переменной d равно... Варианты ответов: 1) 0 2) 10 3) 20 4) 40	06 17	В представленном фрагменте программы $b=11; d=46$ нц пока $d \geq b$ $ d=d-b$ кц тело цикла выполнится... Варианты ответов: 1) 2 раза 2) 1 раз 3) 4 раза 4) 3 раза

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
07 18	В представленном фрагменте программы $b=12; d=46$ нц пока $d \geq b$ $d=d-b$ кц тело цикла выполнится... Варианты ответов: 1) 4 раза 3) 1 раз 2) 2 раза 4) 3 раза	09 00	В представленном фрагменте программы $b=10; d=40$ нц пока $d \geq b$ $d=d-b$ кц тело цикла выполнится... Варианты ответов: 1) 3 раза 3) 2 раза 2) 4 раза 4) 1 раз
08 19	В представленном фрагменте программы $b=10; d=30$ нц пока $d \geq b$ $d=d-b$ кц тело цикла выполнится... Варианты ответов: 1) 3 раза 3) 2 раза 2) 0 раз 4) 1 раз	10 16	После выполнения следующего алгоритма $b=10; d=40$ нц пока $d \geq b$ $d=d-b$ кц значение переменной d равно... Варианты ответов: 1) 0 3) 10 2) 40 4) 30

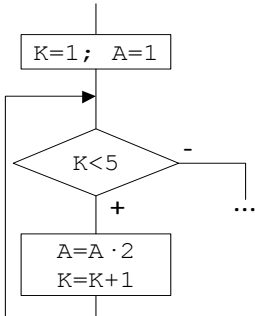
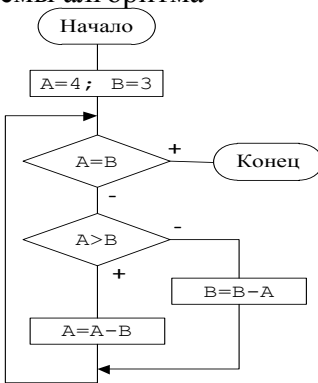
ЗАДАНИЕ 5

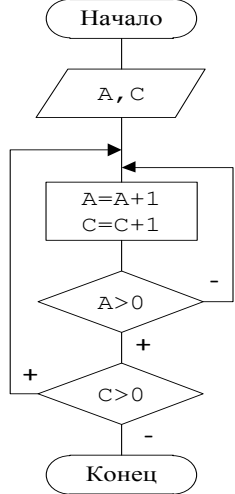
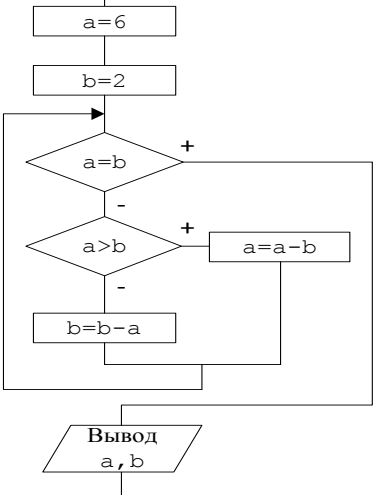
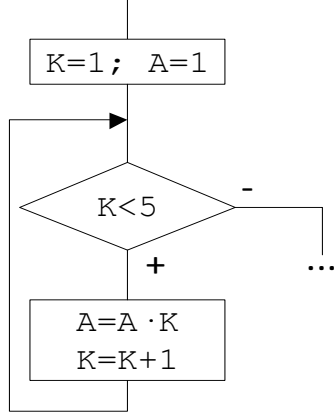
Выбрать правильный ответ из списка предложенных ответов.

В отчете привести: номер правильного ответа и обоснование вашего выбора в свободной форме.

Внимание! Прежде, чем приступить к выполнению данного задания, решите как минимум примеры 11 – 14, 19.

ВАРИАНТЫ ЗАДАНИЯ 5

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
01 11	Представленный фрагмент блок-схемы алгоритма вычисляет...  Варианты ответов: 1) 2^4 3) $1*2*3*4$ 2) 2^5 4) $1*2*3*4*5$	02 12	В результате работы следующей блок-схемы алгоритма  А и В примут значения... Варианты ответов: 1) $A=0; B=0$ 3) $A=3; B=3$ 2) $A=1; B=1$ 4) $A=4; B=3$

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
03 13	<p>В представленном фрагменте программы цикл выполнится _____ раз.</p> <pre> a=3; b=7; ПОКА (a/2) ≤ (b/3) НЦ a=a+2; b=b+3; КЦ; </pre> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) 10 раз 2) 100 раз 3) Бесконечное число раз 4) 1000 раз 	05 15	<p>Алгоритм, представленный блок-схемой,</p>  <p>закончит работу при начальных значениях:</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) A=-4; C=-3 2) A=-3; C=-2 3) A=-2; C=-1 4) A=-2; C=-3
04 14	<p>В результате выполнения следующего фрагмента блок-схемы алгоритма</p>  <p>а и b примут значения...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) a=0; b=0 2) a=2; b=2 3) a=4; b=2 4) a=2; b=4 	06 16	<p>Представленный фрагмент блок-схемы алгоритма вычисляет...</p>  <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) A⁴ 2) 1*2*3*4*5 3) A⁵ 4) 1*2*3*4

Но-	Задание	Но-	Задание
-----	---------	-----	---------

мер ва- ри- анта		мер ва- ри- анта	
07 17	<p>В результате выполнения следующего фрагмента блок-схемы алгоритма</p> <pre> graph TD Start((...)) --> X0[X=0] X0 --> Y0[Y=0] Y0 --> Cond{Y ≤ X ≤ 4} Cond -- "+" --> Xplus[X=X+1] Xplus --> Ycalc[Y=X · X/2] Ycalc --> Cond Cond -- "-" --> End1((...)) End1 --> Out[/Вывод X, Y/] Out --> End2((...)) </pre> <p>X и Y примут следующие значения...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) X=2; Y=2 2) X=3; Y=0,5 3) X=2; Y=3,5 4) X=3; Y=4,5 	09 19	<p>Задан фрагмент алгоритма:</p> <p>если $a < b$, то $c = b - a$, иначе $c = 2 * (a - b)$ $d = 0$ пока $c > a$ выполнить действия $d = d + 1$, $c = c - 1$</p> <p>В результате выполнения данного алгоритма с начальными значениями $a=8$, $b=3$, переменные c и d примут значения...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) $c = 5$; $d = 0$ 2) $c = -5$; $d = 1$ 3) $c = 10$; $d = 1$ 4) $c = 8$; $d = 2$
08 18	<p>После выполнения следующего алгоритма</p> <p>$b = 11$ $d = 46$ нц пока $d \geq b$ $d = d - b$ кц</p> <p>значение переменной d равно...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) 24 2) 35 3) 13 4) 2 	10 00	<p>Данная блок-схема программы...</p> <pre> graph TD Start([Начало]) --> Input[/Ввод X/] Input --> Init[N=1 Y=0] Init --> Cond{N < 10} Cond -- "+" --> Body1[Y = Y + X + N - 1] Body1 --> Body2[N = N + 1] Body2 --> Cond Cond -- "-" --> Output[/Вывод Y/] Output --> End([Конец]) </pre> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Производит сложение 10 подряд идущих натуральных чисел, начиная с введенного, и выводит результат 2) Производит сложение 9 подряд идущих натуральных чисел, начиная с введенного, и выводит результат 3) Возводит введенное число в 9-ю степень и выводит результат 4) Возводит введенное число в 10-ю степень и выводит результат

ЗАДАНИЕ 6

Выбрать правильный ответ из списка предложенных ответов.

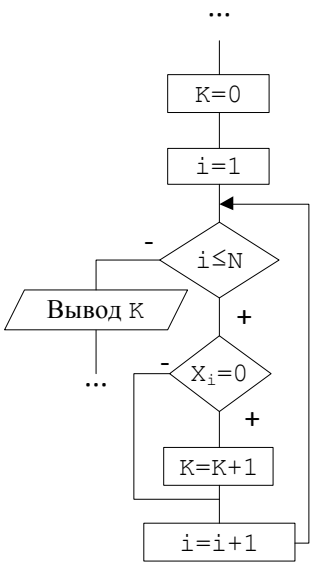
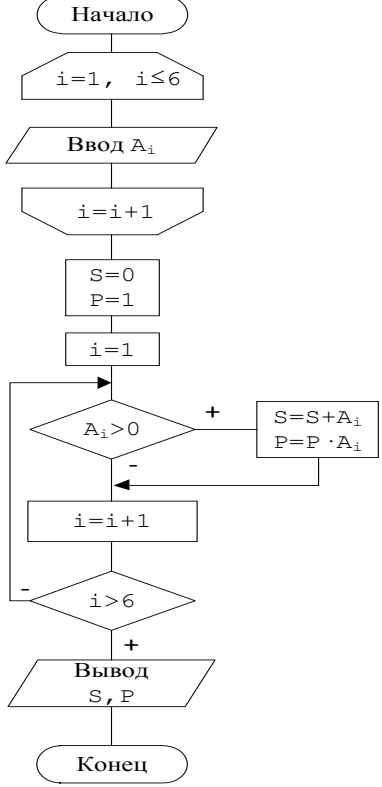
В отчете привести: номер правильного ответа и обоснование вашего выбора в свободной форме.

Внимание! Прежде, чем приступить к выполнению данного задания, решите как минимум примеры 14, 19, 20, 25.

ВАРИАНТЫ ЗАДАНИЯ 6

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
01 16	<p>Укажите пропущенный фрагмент в алгоритме, определяющем количество нечетных элементов в массиве $A[1:N]$.</p> <pre> S=0; K=1 нц для J от 1 до N если _____ то S=S+1 все кц </pre> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) $\text{MOD}(A[J], 2) = K$ 2) $\text{DIV}(A[J], 2) = 1$ 3) $\text{MOD}(A[K], 2) = K$ 4) $\text{MOD}(A[J], 2)$ <p>Примечание. Операция $\text{MOD}(x, y)$ – получение остатка целочисленного деления x на y. Операция $\text{DIV}(x, y)$ – целочисленное деление x на y.</p>	03 18	<p>Дан фрагмент алгоритма, в котором обрабатывается массив $A = (2, 12, 0, -3, 0)$:</p> <pre> цел таб A[1:5] Y=0 нц для k от 1 до 5 если A[k] >= 0 то Y=Y+1 все кц вывод Y </pre> <p>Укажите значение переменной Y после окончания работы.</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) 4 2) 1 3) 11 4) 5
02 17	<p>В результате выполнения фрагмента следующего алгоритма</p> <pre> ... i=2 нц для i от 2 до N A_i = i · i i = i + 2 кц </pre> <p>элементы массива A_2, A_4, A_6, A_8 при $N=8$ получают, соответственно, значения...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) 4, 16, 32, 48 2) 4, 16, 36, 64 3) 4, 12, 24, 36 4) 2, 4, 16, 32 	04 19	<p>Задан одномерный массив X_1, X_2, \dots, X_N. Следующий фрагмент алгоритма определяет...</p> <pre> ... S=0 i=1 нц для i от 1 до N если X_i > 0 то S=S+X_i все i=i+1 кц </pre> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) Максимальный элемент массива 2) Сумму положительных элементов 3) Количество положительных элементов 4) Индекс последнего положительного элемента

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
05 00	<p>Дан массив целых чисел A_i, где $i=1, 2, 3, \dots, M$. Пусть M равно 15. Программа вычисляет произведение сумм некоторых элементов этого массива.</p> <pre> ПРОГРАММА 15; ФУНКЦИЯ SUMMA(I1, I2); НАЧАТЬ ФУНКЦИЮ S=0; НЦ ДЛЯ I=I1 ДО I2 S=S+A[I] КЦ; SUMMA=S КОНЕЦ ФУНКЦИИ; НАЧАТЬ ПРОГРАММУ ПИСАТЬ ("ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА A:"); НЦ ДЛЯ J=1 ДО M ЧИТАТЬ (A[J]); КЦ; P=SUMMA(G, W) * SUMMA(T, L); ПИСАТЬ ("ПРОИЗВЕДЕНИЕ РАВНО:", P) КОНЕЦ ПРОГРАММЫ. </pre> <p>В программе введены следующие константы: $G=1$; $W=12$; $T=8$; $L=15$. Работу программы описывает алгебраическое выражение...</p> <p>Варианты ответов:</p> <p>1) $P = \sum_{i=1}^{12} A_i \cdot \sum_{j=8}^{12} A_j$ 3) $P = \sum_{i=1}^{15} A_i \cdot \sum_{j=1}^6 A_j$ 2) $P = \sum_{i=1}^{12} A_i \cdot \sum_{j=8}^{15} A_j$ 4) $P = \sum_{i=1}^8 A_i \cdot \sum_{j=1}^{12} A_j$</p>	07 12	<p>В результате выполнения следующего фрагмента алгоритма</p> <pre> ... i=1 ↓ i ≤ N / \ + - A_i = i · 2 + 2 i = i + 1 ↓ ... </pre> <p>Элементы массива A_1, A_2, A_3, A_4 при $N=4$ получают, соответственно, значения...</p> <p>Варианты ответов:</p> <p>1) 2, 4, 8, 12 2) 4, 6, 12, 14 3) 2, 8, 16, 32 4) 4, 6, 8, 10</p>
06 11	<p>Задан двумерный массив</p> $A = \begin{pmatrix} 5 & 8 \\ 9 & 6 \end{pmatrix}.$ <p>После выполнения следующего фрагмента алгоритма</p> <pre> V=0 нц для I от 1 до 2 нц для J от I до 2 если A[I, J] > V то V=A[I, J] все кц кц </pre> <p>переменная V примет значение...</p> <p>Варианты ответов:</p> <p>1) 5 2) 6 3) 8 4) 9</p>	08 13	<p>Задан одномерный массив X_1, X_2, \dots, X_N. Следующий фрагмент алгоритма определяет...</p> <pre> ... R=1 i=1 ↓ i ≤ N / \ + - X_i < 0 / \ + - R = R · X_i i = i + 2 ↓ ... </pre> <p>Варианты ответов:</p> <p>1) Произведение отрицательных элементов 2) Произведение положительных элементов с четными номерами 3) Произведение отрицательных элементов с не четными номерами 4) Количество положительных элементов с четными номерами</p>

Но- мер ва- ри- анта	Задание	Но- мер ва- ри- анта	Задание
09 14	<p>Задан одномерный массив X_1, X_2, \dots, X_N. Следующий фрагмент алгоритма определяет...</p>  <pre> graph TD Start((...)) --> K0[K=0] K0 --> i1[i=1] i1 --> iN{i ≤ N} iN -- + --> Xi0{X_i = 0} Xi0 -- + --> Kinc[K=K+1] Kinc --> iinc[i=i+1] iinc --> iN iN -- - --> OutK[/Вывод K/] OutK --> End((...)) </pre> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) номер последнего нулевого элемента 2) номер первого нулевого элемента 3) количество нулевых элементов 4) количество положительных элементов 	10 15	<p>Результатом выполнения алгоритма, представленного следующей блок-схемой</p>  <pre> graph TD Start([Начало]) --> i1i6{i=1, i ≤ 6} i1i6 --> InAi[/Ввод A_i/] InAi --> iinc[i=i+1] iinc --> S0P1[S=0 P=1] S0P1 --> i1[i=1] i1 --> AiGT{A_i > 0} AiGT -- + --> SPS[P=S+A_i P=P·A_i] SPS --> iinc2[i=i+1] AiGT -- - --> iinc2 iinc2 --> iGT6{i > 6} iGT6 -- + --> OutSP[/Вывод S, P/] OutSP --> End([Конец]) iGT6 -- - --> i1 </pre> <p>для массива чисел (1, -5, -9, 2, -10, 3), являются значения...</p> <p>Варианты ответов:</p> <ol style="list-style-type: none"> 1) $S=-24$; $P=-450$ 2) $S=24$; $P=-250$ 3) $S=-19$; $P=-270$ 4) $S=6$; $P=6$

ЗАДАНИЕ 7

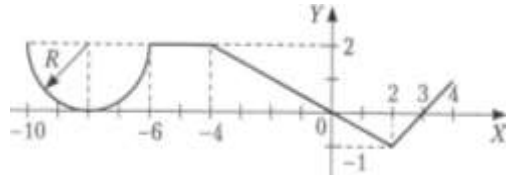
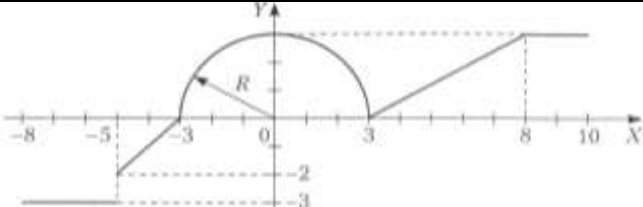
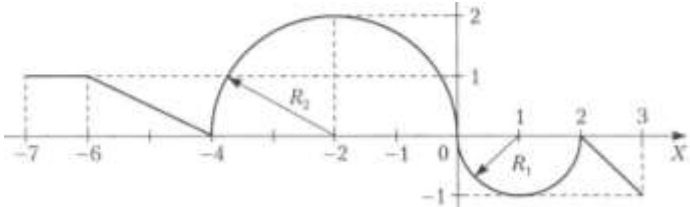
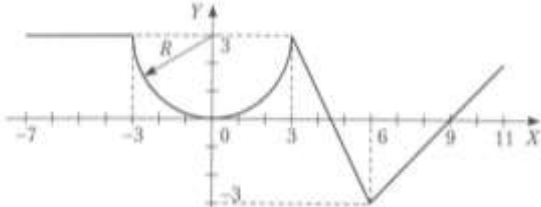
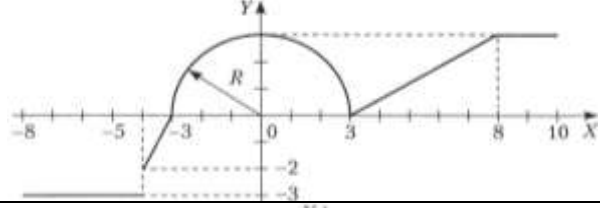
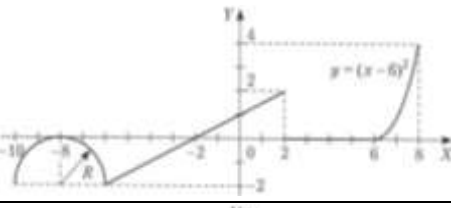
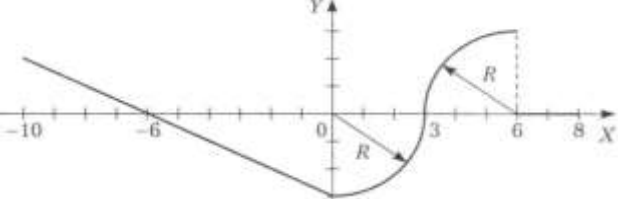
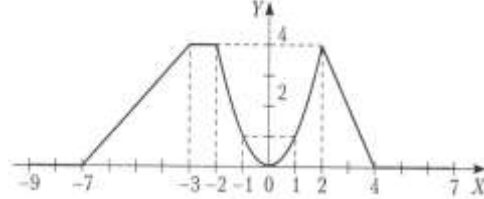
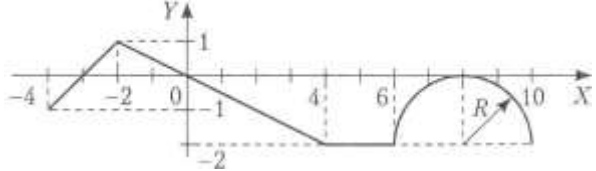

Составить алгоритм для вычисления значения функции, заданной в виде графика, по введенному значению аргумента. Параметр R ввести с клавиатуры.

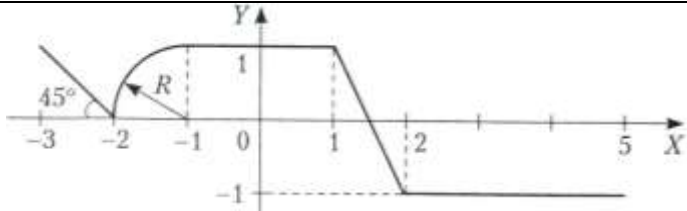
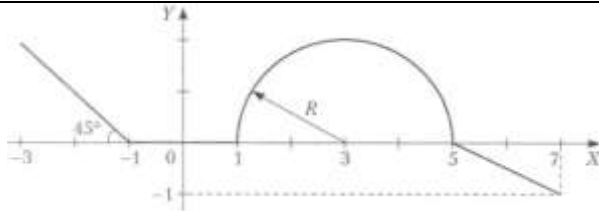
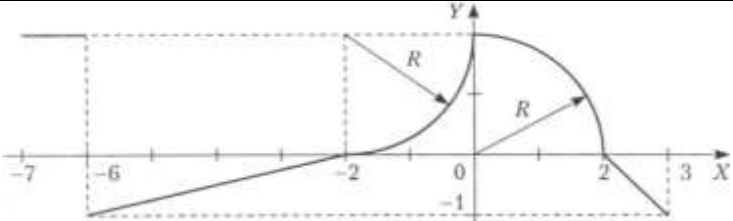
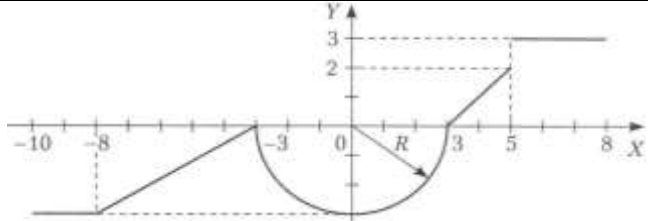
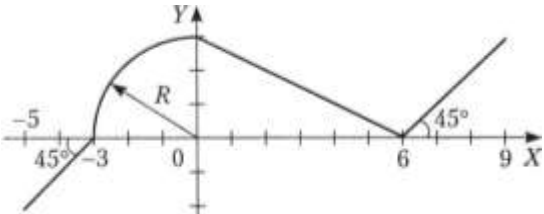
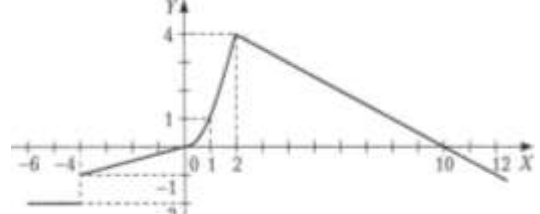
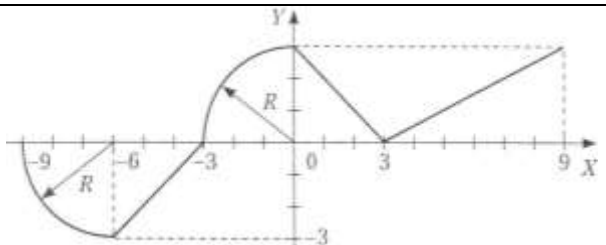
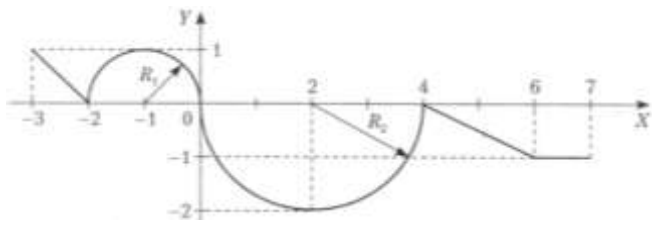
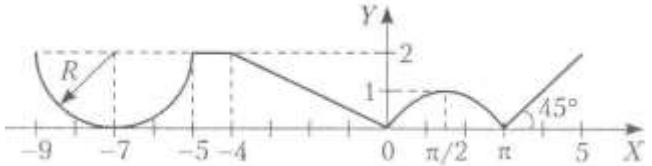
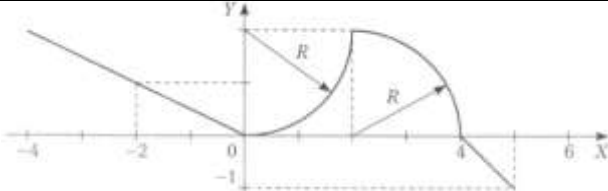
В отчете привести: постановку задачи, математическую модель, алгоритм в виде блок-схемы с подробными комментариями.

Примечание. В алгоритме отделить ввод данных, вычисление функции и вывод значений аргумента и функции. Учесть, что функция определена не на всей числовой оси.

Внимание! Прежде, чем приступать к выполнению данного задания, прочитайте как минимум подразд. 1.5, 1.6.2 и решите как минимум примеры 6, 7, 8.

ВАРИАНТЫ ЗАДАНИЯ 7

Номер варианта	Задание	Номер варианта	Задание
00		05	
01		06	
02		07	
03		08	
04		09	

Номер варианта	Задание	Номер варианта	Задание
10		15	
11		16	
12		17	
13		18	
14		19	

ЗАДАНИЕ 8

Составить алгоритм для определения, попадает ли точка с произвольно заданными координатами (x, y) в область, закрашенную на рисунке серым цветом. Координаты точки (x, y) и другие необходимые данные задать самостоятельно.

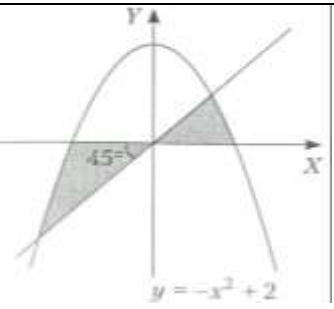
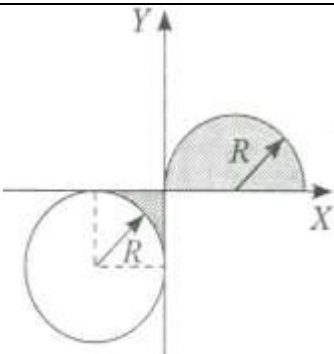
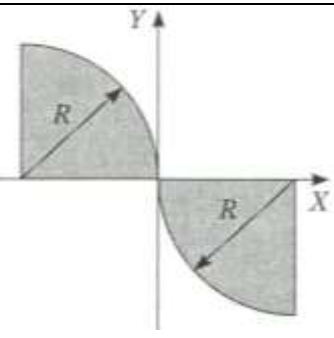
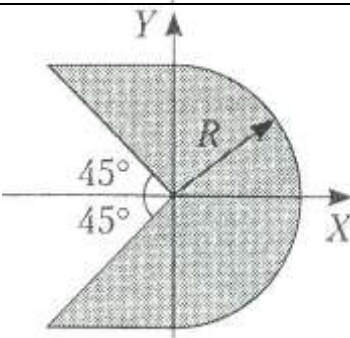
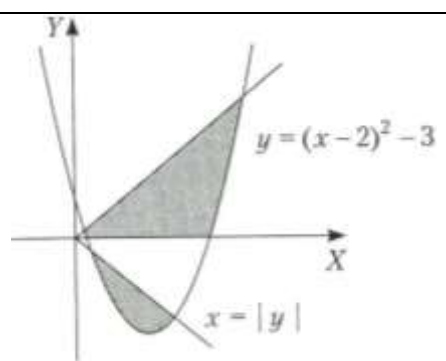
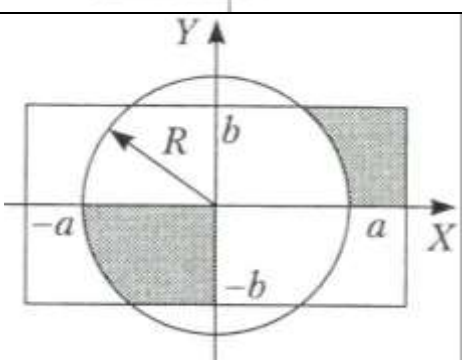
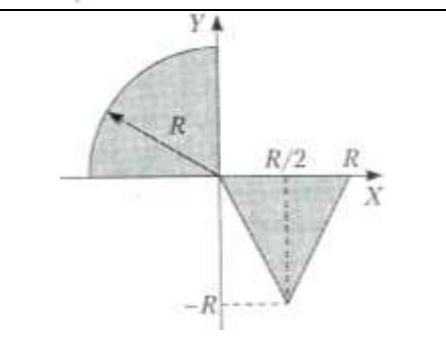
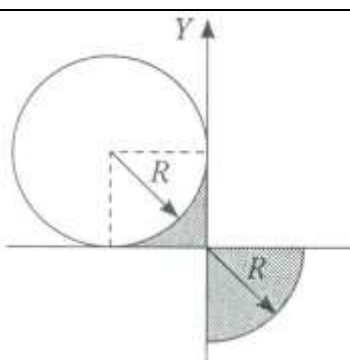
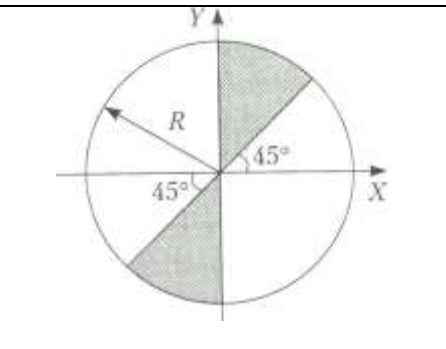
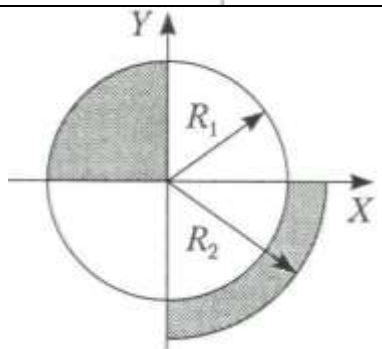
В отчете привести: постановку задачи, математическую модель, алгоритм в виде блок-схемы с подробными комментариями.

Примечание. В алгоритме отделить ввод данных, определение попадания точки в область и вывод соответствующего сообщения.

Внимание! Прежде, чем приступать к выполнению данного задания, прочитайте как минимум подразд. 1.5, 1.6.2 и решите как минимум примеры 6 - 8.

ВАРИАНТЫ ЗАДАНИЯ 8

Номер варианта	Задание	Номер варианта	Задание
00		04	
01		05	
02		06	
03		07	

Номер варианта	Задание	Номер варианта	Задание
08		13	
09		14	
10		15	
11		16	
12		17	

Номер варианта	Задание	Номер варианта	Задание
18		19	

ЗАДАНИЕ 9

Даны действительные числа x , ε ($x \neq 0$, $\varepsilon > 0$). Разработать алгоритм вычисления суммы ряда с точностью ε и подсчета количества слагаемых. Недостающие данные задать самостоятельно.

В отчете привести: постановку задачи, математическую модель, расчеты вычисления дополнения и первого члена ряда, алгоритм в виде блок-схемы с подробными комментариями, «ручную отладку» алгоритма.

Внимание! Прежде, чем приступать к выполнению данного задания, прочитайте как минимум подразд. 1.5, 1.6.3 и решите как минимум пример 16.

ВАРИАНТЫ ЗАДАНИЯ 9

Номер варианта	Задание
00	$\sum_{k=0}^{\infty} \frac{x^{2 \cdot k}}{2^k \cdot k!}$
01	$\sum_{k=0}^{\infty} \frac{(-1)^{k+1}}{(2 \cdot k)!} \left(\frac{x}{3}\right)^{4 \cdot k}$
02	$\sum_{k=0}^{\infty} \frac{(-1)^k \cdot x^{2 \cdot k+1}}{k! \cdot (2 \cdot k + 1)}$
03	$\sum_{k=0}^{\infty} \frac{(-1)^k \cdot x^{k+2}}{(k+2)! \cdot (k+1)}$
04	$\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{(2 \cdot k + 1)!} \cdot \left(\frac{x}{3}\right)^{4 \cdot k+2}$
05	$\sum_{k=0}^{\infty} \frac{(-1)^k \cdot x^{4 \cdot k+1}}{(2 \cdot k)! \cdot (4 \cdot k + 1)}$
06	$\sum_{k=0}^{\infty} \frac{(-1)^k}{((k+1)!)^2} \cdot \left(\frac{x}{2}\right)^{2 \cdot (k+1)}$
07	$\sum_{k=0}^{\infty} \frac{(-1)^k}{k! \cdot (k+n)!} \cdot \left(\frac{x}{2}\right)^{n+2 \cdot k}$
08	$1 - 2 \cdot x + \frac{4 \cdot x^2}{2!} - \dots + (-1)^n \cdot \frac{2^n x^n}{n!} + \dots$
09	$1 + 3 \cdot x^2 + \frac{5 \cdot x^4}{2!} + \dots + \frac{2 \cdot n + 1}{n!} x^{2 \cdot n} + \dots$

Номер варианта	Задание
10	$3 \cdot x + \frac{8 \cdot x^2}{2!} + \frac{15 \cdot x^3}{3!} + \dots + \frac{n \cdot (n+2) \cdot x^n}{n!} + \dots$
11	$\frac{x-1}{x+1} + \frac{1}{3} \cdot \left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5} \cdot \left(\frac{x-1}{x+1}\right)^5 + \dots + \frac{1}{2 \cdot n + 1} \cdot \left(\frac{x-1}{x+1}\right)^{2 \cdot n + 1} + \dots$
12	$\sum_{k=0}^{\infty} \frac{(-1)^k}{k! \cdot (k+1)!} \cdot \left(\frac{x}{2}\right)^{2 \cdot k + 1}$
13	$\sum_{k=0}^{\infty} \frac{(-1)^k \cdot x^{4 \cdot k + 3}}{(2 \cdot k + 1)! \cdot (4 \cdot k + 3)}$
14	$\sum_{k=0}^{\infty} \frac{(-1)^{k+1} \cdot x^{2 \cdot k - 1}}{(2 \cdot k + 1)! \cdot (2 \cdot k - 1)}$
15	$1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots$
16	$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^{2 \cdot n}}{(2 \cdot n)!} + \dots$
17	$1 + \sum_{k=1}^{\infty} \frac{a \cdot (a-1) \cdot \dots \cdot (a-k+1)}{k!} \cdot x^k$
18	$x - \frac{x^3}{3} + \frac{x^5}{5} - \dots + (-1)^n \cdot \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} + \dots$
19	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^n \cdot \frac{x^{2 \cdot n}}{(2 \cdot n)!} + \dots$

ЗАДАНИЕ 10

Разработать циклический алгоритм вычисления заданного выражения.

В отчете привести: постановку задачи, математическую модель, алгоритм в виде блок-схемы с подробными комментариями, «ручную отладку» алгоритма.

Примечание. Не использовать массивы.

Внимание! Прежде, чем приступать к выполнению данного задания, прочитайте как минимум подразд. 1.5, 1.6.3 и решите как минимум пример 15.

ВАРИАНТЫ ЗАДАНИЯ 10

Номер варианта	Задание
00	<p>Вычислить:</p> $1 + \frac{1}{3 + \frac{1}{5 + \frac{1}{\dots + \frac{1}{101 + \frac{1}{103}}}}}$
01	<p>Задано натуральное число n. Вычислить: $\sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}$.</p> <p>Примечание: вычисляется n корней.</p>

Номер варианта	Задание
02	Дано натуральное число n и действительное x . Вычислить: $\sum_{i=1}^n \frac{x + \cos(i \cdot x)}{2^i}.$
03	Дано натуральное число n , действительное число a . Вычислить: $\sum_{i=0}^n \frac{(-1)^i}{a^{2i}}.$
04	Пусть: $x_1=y_1=1$; $x_i=0,3 \cdot x_{i-1}$; $y_i=x_{i-1}+y_{i-1}$; $i=1, 2, 3 \dots$ Дано натуральное число n . Найти: $\sum_{i=1}^n i! \cdot \frac{x_i}{1 + y_i }.$
05	Дано натуральное число n и действительные числа u и v . Пусть $a_1=u$; $b_1=v$; $a_k=2 \cdot b_{k-1}+a_{k-1}$; $b_k=2 \cdot a_{k-1}+b_{k-1}$, $k=2, 3, \dots$ Вычислить: $\sum_{k=1}^n \frac{a_k \cdot b_k}{(k+1)!}.$
06	Дано натуральное число n , действительные числа x , a . Вычислить: $\sum_{i=0}^n \frac{1}{a \cdot (a+x) \cdot \dots \cdot (a+x^i)}.$
07	Дано натуральное число n . Вычислить: $\sqrt{3 + \sqrt{6 + \dots + \sqrt{3 \cdot (n-1) + \sqrt{3 \cdot n}}}}.$
08	Дано натуральное число n . Вычислить: $\sum_{i=1}^n \sqrt{3 \cdot i + \sqrt{3 \cdot (i-1) + \dots + \sqrt{6 + \sqrt{3}}}}.$
09	Дано натуральное число n . Вычислить: $\frac{\cos 1}{\sin 1} \cdot \frac{\cos 1 + \cos 2}{\sin 1 + \sin 2} \cdot \dots \cdot \frac{\cos 1 + \dots + \cos n}{\sin 1 + \dots + \sin n}.$
10	Дано натуральное число n . Вычислить: $\sum_{i=1}^n \frac{\cos(1)}{\sin(1)} \cdot \frac{\cos(1) + \cos(2)}{\sin(1) + \sin(2)} \cdot \dots \cdot \frac{\cos(1) + \dots + \cos(i)}{\sin(1) + \dots + \sin(i)}.$
11	Дано натуральное число n , действительные числа x , a . Вычислить: $((\dots((x+a)^2 + a)^2 + \dots + a)^2 + a).$ Примечание: сразу открывается, а затем постепенно закрывается n скобок.
12	Дано натуральное число n , действительные числа a и x . Вычислить: $\sum_{i=0}^n ((\dots((x+a)^2 + a)^2 + \dots + a)^2 + a).$ Примечание: после знака суммы стоит i скобок.
13	Дано натуральное число n , действительное число x . Вычислить: $\sum_{i=1}^n \frac{(x-2) \cdot (x-4) \cdot (x-8) \cdot \dots \cdot (x-2^i)}{(x-1) \cdot (x-3) \cdot (x-7) \cdot \dots \cdot (x-2^i+1)}.$

Номер варианта	Задание
14	Вычислить: $x^2 + \frac{\frac{x}{2}}{x^2 + \frac{4}{x^2 + \frac{8}{\dots + x^2 + \frac{256}{x^2}}}}$
15	Дано натуральное число n . Вычислить: $\sum_{i=1}^n 2^{i!}$.
16	Дано натуральное число n и действительное x . Вычислить: $\prod_{k=1}^n \left(1 + \frac{\sin(k \cdot x)}{k!}\right).$
17	Дано натуральное число n . Вычислить: $\sum_{i=1}^n \left(1 + \frac{1}{1^2}\right) \cdot \left(1 + \frac{1}{2^2}\right) \cdot \dots \cdot \left(1 + \frac{1}{i^2}\right).$
18	Дано натуральное число n . Вычислить: $\sum_{k=1}^n \frac{k!}{\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k+1}}.$
19	Пусть $a_1=b_1=1$; $a_k=3 \cdot b_{k-1}+2 \cdot a_{k-1}$; $b_k=2 \cdot a_{k-1}+b_{k-1}$, $k=2, 3, \dots$ Дано натуральное число n . Вычислить: $\sum_{k=1}^n \frac{2^k}{(1 + a_k^2 + b_k^2) \cdot k!}.$

ЗАДАНИЕ 11

Разработать алгоритм для работы с цифрами натурального числа. Предусмотреть печать заданного числа и всех результатов.

В отчете привести: постановку задачи, математическую модель, тестовые примеры для более наглядного представления условия задачи и алгоритма, алгоритм в виде блок-схемы с подробными комментариями, «ручную отладку» алгоритма.

Примечание. Не использовать строковые функции.

Внимание! Прежде, чем приступить к выполнению данного задания, решите как минимум примеры 17, 18.

ВАРИАНТЫ ЗАДАНИЯ 11

Номер варианта	Задание
00	Для заданного натурального числа n определить сумму цифр кратных трем.
01	Найти сумму цифр младшего и старшего разрядов заданного натурального числа k .
02	Задано натуральное число n . Найти количество цифр кратных четырем.
03	Определить сумму цифр нечетных разрядов заданного натурального числа m .
04	Задано натуральное число n . Найти номер самого старшего разряда с цифрой кратной трем.

Номер варианта	Задание
05	Определить сумму цифр четных разрядов заданного натурального числа n .
06	Определить произведение и количество цифр заданного натурального числа k .
07	Для заданного натурального числа m определить номер самого младшего разряда с цифрой кратной четырем.
08	Задано натуральное число k . Найти произведение цифр кратных четырем.
09	Определить количество цифр нечетных разрядов заданного натурального числа n .
10	Найти количество цифр четных разрядов заданного натурального числа m .
11	Для заданного натурального числа k определить произведение его цифр кратных трем.
12	Задано натуральное число n . Найти сумму его цифр кратных четырем.
13	Для заданного натурального числа k определить количество цифр кратных трем.
14	Задано натуральное число m . Найти номер самого старшего разряда с цифрой кратной четырем.
15	Найти произведение цифр нечетных разрядов заданного натурального числа k .
16	Для заданного натурального числа m определить номер самого младшего разряда с цифрой кратной трем.
17	Определить произведение цифр четных разрядов заданного натурального числа k .
18	Найти сумму и количество цифр заданного натурального числа n .
19	Найти произведение цифр старшего и младшего разрядов заданного натурального числа m .

ЗАДАНИЕ 12

Разработать алгоритм обработки одномерных числовых массивов. Размер и значения элементов исходного массива задать самостоятельно, предусмотреть печать исходных данных и всех результатов.

В отчете привести: постановку задачи, математическую модель, тестовые примеры для более наглядного представления условия задачи и алгоритма, алгоритм в виде блок-схемы с подробными комментариями, «ручную отладку» алгоритма.

Примечание. Порядок следования чисел в новом массиве должен быть сохранен таким же, как в исходном массиве.

Внимание! Прежде, чем приступить к выполнению данного задания, прочитайте как минимум подразд. 1.7 и решите как минимум примеры 14, 20, 21, 22, 25, 31.

ВАРИАНТЫ ЗАДАНИЯ 12

Номер варианта	Задание
00	Задана последовательность чисел b . Сформировать новую последовательность a из четных чисел последовательности b , расположенных на нечетных местах после последнего по порядку минимального числа
01	Сформировать новый массив c из четных чисел заданного массива a , расположенных после последнего по порядку минимального числа
02	Переписать в новый массив a числа заданного массива d , расположенные до первого нечетного числа. В новом массиве a найти наименьший номер максимального значения

Номер варианта	Задание
03	Нечетные числа заданного массива d , расположенные до первого по порядку минимального числа, переписать в новый массив b
04	Из нечетных чисел заданного массива c , расположенных на нечетных местах после последнего по порядку максимального числа, сформировать новый массив b
05	Переписать в новый массив b сначала четные, затем нечетные числа заданного массива d . В новом массиве b найти наибольший номер максимального значения
06	Сформировать новую последовательность d из чисел с нечетными номерами, расположенных после последнего по порядку минимального числа в исходной последовательности c
07	Числа заданного массива b , расположенные на нечетных местах до первого по порядку максимального числа, переписать в новый массив d
08	Сформировать новую последовательность d из чисел, расположенных сначала на нечетных местах, а затем на четных местах в исходной последовательности a . В новой последовательности d найти наименьший номер максимального значения
09	Из заданного массива c переписать в новый массив a числа, расположенные на четных местах до первого по порядку минимального числа
10	Сформировать новый массив c из чисел заданного массива d , расположенных после первого нечетного числа. В исходном массиве d найти наибольший номер минимального значения
11	Из четных чисел заданного массива a , расположенных до первого по порядку максимального числа, сформировать новый массив b
12	Сформировать новый массив b из чисел заданного массива c , расположенных до первого четного числа. В новом массиве b найти наименьший номер минимального значения
13	Переписать в новый массив c нечетные числа заданного массива a , расположенные после последнего по порядку максимального числа
14	Из чисел заданного массива b , расположенных после первого четного числа, сформировать новый массив c . В исходном массиве b найти наибольший номер максимального значения
15	Сформировать новую последовательность d из нечетных чисел с четными номерами, расположенных до первого по порядку минимального числа в исходной последовательности a
16	Задана последовательность чисел b . Сформировать новую последовательность c из чисел последовательности b , расположенных на четных местах после последнего по порядку максимального числа
17	Из заданного массива c переписать в новый массив d сначала нечетные, затем четные числа. В исходном массиве c найти наименьший номер минимального значения
18	Задана последовательность чисел d . Найти в ней наибольший номер минимального значения. Сформировать новую последовательность c из чисел, расположенных сначала на четных местах, а затем на нечетных местах в исходной последовательности d
19	Из заданного массива b переписать в новый массив d четные числа, расположенные на четных местах до первого по порядку максимального числа

ЗАДАНИЕ 13

Разработать алгоритм обработки одномерных числовых массивов. Размер и значения элементов исходного массива задать самостоятельно, предусмотреть печать значений элементов исходного и результирующего массивов.

В отчете привести: постановку задачи, математическую модель, тестовые примеры для более наглядного представления условия задачи и алгоритма, алгоритм в виде блок-схемы с подробными комментариями, «ручную отладку» алгоритма.

Примечание. Не использовать новый массив.

Внимание! Прежде чем приступить к выполнению данного задания, прочитайте как минимум подразд. 1.7 и решите как минимум примеры 14, 23, 24, 26.

ВАРИАНТЫ ЗАДАНИЯ 13

Номер варианта	Задание
00	В заданном одномерном массиве b удалить все элементы, равные заданному значению a
01	В заданной последовательности чисел a удалить все числа, принадлежащие заданному интервалу значений $[b; c]$
02	Расширить заданный одномерный массив c , вставив заданный элемент a до элемента с указанным номером m
03	Преобразовать заданную последовательность чисел c путем удаления из нее всех чисел больше заданного значения a
04	Сжать заданную последовательность чисел a , удалив из нее все отрицательные элементы
05	Преобразовать заданный одномерный массив b путем вставки в него заданного элемента a после всех отрицательных элементов
06	Расширить заданную последовательность чисел d , вставив в нее заданное число a перед каждым нулем
07	В заданную последовательность чисел d вставить заданный элемент a перед каждым положительным
08	Сжать заданный одномерный массив a , удалив из него все отрицательные числа с номерами от k до m
09	Преобразовать заданную последовательность чисел d путем вставки в нее заданного числа a после всех чисел, равных указанному числу b
10	В заданном одномерном массиве a удалить все нулевые элементы
11	В заданный одномерный массив b вставить заданный элемент a после элемента с указанным номером m
12	Сжать заданный одномерный массив b , удалив из него все элементы, меньшие заданного значения a
13	Расширить заданную последовательность чисел c , вставив в нее заданное число a перед каждым числом, равным заданному числу b
14	Преобразовать заданный одномерный массив d путем удаления из него всех положительных элементов
15	Преобразовать заданную последовательность чисел c путем удаления из нее всех положительных чисел с номерами от k до m
16	В заданной последовательности чисел a удалить все нули с номерами от k до m

Номер варианта	Задание
17	В заданный одномерный массив c вставить заданный элемент a после каждого отрицательного элемента
18	Расширить заданный одномерный массив b , вставив заданный элемент a до каждого положительного элемента
19	В заданную последовательность чисел d вставить заданный элемент a перед каждым элементом из заданного интервала значений $[b; c]$

ЗАДАНИЕ 14

Разработать алгоритм обработки двумерных числовых массивов.

Размер и значения элементов исходного массива задать самостоятельно, предусмотреть печать значений элементов исходного и результирующего массивов и всех полученных результатов.

В отчете привести: постановку задачи, математическую модель, тестовые примеры для более наглядного представления условия задачи и алгоритма, алгоритм в виде блок-схемы с подробными комментариями, «ручную отладку» алгоритма.

Примечание. Средним геометрическим n положительных чисел является корень n степени из произведения этих чисел. Для обращения к элементам массива, находящимся в определенных областях массива, циклы организовывать так, чтобы это обращение происходило заведомо только к нужным элементам массива, то есть не использовать лишних проверок для индексов элементов массива.

Внимание! Прежде чем приступить к выполнению данного задания, прочитайте как минимум подразд. 1.7, 1.7.2, 1.7.3 и решите как минимум примеры 19, 20, 27 – 32.

ВАРИАНТЫ ЗАДАНИЯ 14

Номер варианта	Задание
00	В матрице a определить среднее геометрическое элементов, расположенных ниже побочной диагонали. Пузырьковой сортировкой упорядочить элементы последней строки по убыванию
01	В матрице c определить произведение элементов, находящихся ниже главной диагонали. Расставить элементы последнего столбца по убыванию пузырьковой сортировкой
02	Дана квадратная матрица a нечетного размера n . Найти среднее геометрическое элементов, стоящих не выше главной диагонали. Отсортировать элементы среднего столбца по возрастанию пузырьковой сортировкой
03	В матрице c определить среднее геометрическое элементов, расположенных ниже главной и выше побочной диагоналей. Расположить элементы предпоследней строки по возрастанию методом простого выбора
04	В матрице a найти среднее арифметическое элементов, расположенных выше побочной диагонали. Методом простого выбора упорядочить элементы первой строки по возрастанию
05	Дана квадратная матрица c нечетного размера n . Определить среднее арифметическое элементов, находящихся не ниже главной диагонали. Отсортировать элементы средней строки по возрастанию пузырьковой сортировкой
06	Дана квадратная матрица a нечетного размера n . Определить среднее арифметическое элементов побочной диагонали. Упорядочить элементы среднего столбца по возрастанию методом простого выбора

Номер варианта	Задание
07	Дана квадратная матрица c нечетного размера n . Определить сумму элементов, расположенных не ниже побочной диагонали. Упорядочить элементы среднего столбца по убыванию пузырьковой сортировкой
08	Дана квадратная матрица a нечетного размера n . Найти среднее геометрическое элементов главной диагонали. Отсортировать элементы средней строки по возрастанию методом простого выбора
09	Дана квадратная матрица c четного размера n . Найти среднее геометрическое элементов, расположенных не выше побочной диагонали. Пузырьковой сортировкой расставить элементы первой строки по убыванию
10	В матрице b найти произведение элементов, стоящих выше главной и ниже побочной диагоналей. Расставить элементы второго столбца по убыванию пузырьковой сортировкой
11	В матрице d найти сумму элементов, стоящих выше главной диагонали. Методом простого выбора расставить элементы первого столбца по возрастанию
12	Дана квадратная матрица b нечетного размера n . Найти сумму элементов, стоящих не на главной диагонали. Упорядочить элементы среднего столбца по убыванию методом простого выбора
13	Дана квадратная матрица d нечетного размера n . Определить произведение элементов, находящихся не на побочной диагонали. Пузырьковой сортировкой расставить элементы средней строки по убыванию
14	В матрице b определить среднее арифметическое элементов, расположенных выше главной и выше побочной диагоналей. Упорядочить элементы второй строки по убыванию методом простого выбора
15	Дана квадратная матрица d четного размера n . Определить среднее арифметическое элементов, расположенных не ниже побочной диагонали. Расположить элементы последней строки по возрастанию методом простого выбора
16	Дана квадратная матрица b четного размера n . Найти сумму элементов, стоящих не выше главной диагонали. Отсортировать элементы первого столбца по убыванию пузырьковой сортировкой
17	Дана квадратная матрица d четного размера n . Определить произведение элементов, находящихся не ниже главной диагонали. Методом простого выбора упорядочить элементы последнего столбца по возрастанию
18	В матрице b найти сумму элементов, находящихся ниже главной и ниже побочной диагоналей. Отсортировать элементы предпоследнего столбца по возрастанию пузырьковой сортировкой
19	Дана квадратная матрица d нечетного размера n . Найти произведение элементов, расположенных не выше побочной диагонали. Методом простого выбора расположить элементы средней строки по убыванию

ЛИТЕРАТУРА

1. **Острейковский, В.А.** Информатика: учебник для вузов / В. А. Острейковский. – М.: Высшая школа, 2001, 2000. – 512 с.
2. **Алексеев, А.П.** Информатика 2001 / А.П. Алексеев. – М.: СОЛОН-Р, 2001. – 364 с.
3. **Основы современных компьютерных технологий:** учеб. пособие для вузов / под ред. А.Д. Хомоненко. – 2-е изд. – СПб.: КОРОНА-принт, 2002. – 446 с.
4. **Веретенникова, Е.Г.** Информатика: учеб. пособие для вузов / Е.Г. Веретенникова, С.М. Патрушина, Н.Г. Савельева. – Ростов н/Д: МарТ, 2002. – 413 с.
5. **Могилев, А.В.** Информатика: учеб. пособие для пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; под ред. Е.К. Хеннера. – 2-е изд., стереотип. – М.: Академия, 2003. – 811 с.
6. **Давыдов, В.Г.** Программирование и основы алгоритмизации: учеб. пособие для вузов / В.Г. Давыдов. – М.: Высшая школа, 2003. – 448 с.
7. **Брукшир, Дж.Г.** Информатика и вычислительная техника / Дж. Г. Брукшир. – 7-е изд. – СПб.: Питер, 2004. – 619 с.
8. **ГОСТ 19.701–90.** Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила их выполнения.
9. Информатика: Общий курс: учебник для вузов / А. Н. Гуда, М. А. Бутакова, Н. М. Нечитайло, А. В. Чернов; Под общ.ред. В.И.Колесникова. - М.; Ростов н/Д: Дашков и К; Наука-Пресс, 2007. - 400с.: ил. - Библиогр.: с.391-392. - 201-00.
10. **Новичков, В.С.** Алгоритмизация и программирование на Турбо Паскале: Учебное пособие для вузов / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. - М.: Горячая линия - Телеком, 2005. - 462с. - Библиогр.: с.454. - 215-00.
11. **Берлинер, Э.М.** Microsoft Windows XP / Э. М. Берлинер, И. Б. Глазырина, Б. Э. Глазырин. - 2-е изд., стер. - М.: Бином, 2007. - 510с.: ил. - (К вершинам мастерства). - 245-00.
12. **Мандел, Т.** Дизайн интерфейсов / Т. Мандел; Пер. с англ. - М.: ДМК Пресс, 2005. - 410с.: ил. - (Самоучитель). - Библиогр. в конце глав. - 173-00.

