

ОСНОВЫ АЛГОРИТМИЗАЦИИ

ОМСК 2020

Министерство транспорта Российской Федерации
Федеральное агентство железнодорожного транспорта
Омский государственный университет путей сообщения

ОСНОВЫ АЛГОРИТМИЗАЦИИ

Утверждено методическим советом университета
в качестве учебно-методического пособия
к выполнению самостоятельной работы

Омск 2020

УДК 004.021(075.8)
ББК 32.973я73
О75

Основы алгоритмизации: Учебно-методическое пособие к выполнению самостоятельной работы / Е. А. Сидорова, С. П. Железняк, Т. В. Манохина, С. А. Ступаков; Омский гос. ун-т путей сообщения. Омск, 2020. 36 с.

Учебно-методическое пособие разработано в соответствии с рабочими программами дисциплин информационного профиля с учетом требований федеральных государственных образовательных стандартов высшего образования.

Содержит базовые теоретические и практические сведения об алгоритмизации вычислительных процессов. Приведен перечень свойств и способов представления алгоритмов. Рассмотрены базовые структуры алгоритмов, являющиеся основой построения любой алгоритмической конструкции. Представлены примеры решения задач и индивидуальные задания.

Предназначено для самостоятельной работы студентов всех направлений подготовки (специальностей) очной и заочной форм обучения, изучающих алгоритмизацию вычислительных процессов.

Библиогр.: 3 назв. Табл. 22. Рис. 3.

Рецензенты: доктор техн. наук, профессор В. Н. Горюнов;
доктор техн. наук, профессор В. В. Харламов

ОГЛАВЛЕНИЕ

Введение	5
1. Этапы решения задач на компьютере	6
2. Понятие алгоритма и его свойства	7
3. Способы представления алгоритма.....	8
3.1. Алгоритмический язык (псевдокод).....	9
3.2. Графическая схема алгоритма	10
4. Базовые структуры алгоритмов	12
4.1. Алгоритмы линейной структуры.....	12
4.1.1. Понятие линейного алгоритма	12
4.1.2. Задания	13
4.2. Алгоритмы разветвляющейся структуры	13
4.2.1. Сокращенная форма разветвления	14
4.2.2. Полная форма разветвления.....	15
4.2.3. Задания	16
4.3. Алгоритмы циклической структуры	21
4.3.1. Арифметический цикл.....	21
4.3.2. Итерационный цикл.....	23
4.3.3. Задания	28
5. Контрольные вопросы.....	33
6. Примеры тестовых вопросов.....	33
Библиографический список.....	35

ВВЕДЕНИЕ

Понятие алгоритма является одним из фундаментальных в математике и информатике и применяется для обозначения последовательности действий, приводящей к решению поставленной задачи; оно близко к таким понятиям, как «метод» (например, метод Гаусса для решения систем линейных уравнений) и «способ» (например, способ построения геометрических фигур).

Процесс построения алгоритма решения задачи, результатом которого является выделение этапов обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения, называется *алгоритмизацией*. Алгоритмизация наряду с моделированием выступает в качестве общего метода информатики.

При разработке и записи алгоритма необходимо соблюдать ряд обязательных требований, которые обеспечивают его однозначное понимание, эффективное выполнение и получение определенного результата.

В пособии приведены общие сведения о видах и свойствах алгоритмов, описаны различные способы записи базовых алгоритмических структур, представлены примеры решения задач и индивидуальные задания, контрольные и тестовые вопросы.

Пособие предназначено для студентов первого курса всех направлений подготовки (специальностей) очной и заочной форм обучения, а также может быть использовано для самостоятельного изучения основ алгоритмизации, без знания которых невозможно освоение технологий программирования.

Библиографический список, приведенный в конце пособия, содержит литературу для углубленного изучения материала по рассматриваемой тематике.

1. ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА КОМПЬЮТЕРЕ

В процессе подготовки и решения различных профессиональных задач на компьютере можно выделить следующие этапы.

1. Постановка задачи:

- сбор информации о задаче;
- определение условия задачи;
- выявление конечных целей решения задачи;
- установление формы выдачи результатов;
- описание данных (их типов, диапазонов величин, структуры и т. п.).

2. Анализ, исследование и моделирование задачи:

- исследование существующих аналогов;
- изучение технических и программных средств;
- разработка модели (формализация задачи);
- разработка структур данных.

3. Построение алгоритма:

- выбор метода проектирования алгоритма;
- выявление способа записи алгоритма;
- определение состава тестов и метода тестирования алгоритма;
- разработка алгоритма.

4. Этап программирования:

- определение языка программирования;
- уточнение способов организации данных;
- перевод алгоритма на выбранный язык программирования.

5. Этап тестирования и отладки программы:

- выявление и устранение синтаксических, семантических и логических ошибок в программе;
- тестовые расчеты и анализ результатов тестирования;
- совершенствование полученной программы.

6. Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2 – 5.

7. Сопровождение программы:

- доработка программы для решения конкретных задач;
- составление документации к решенной задаче, математической модели, алгоритму, программе, набору тестов и т. п.

Часто эту последовательность называют *технологической цепочкой решения задачи на ЭВМ*. Однако не все задачи требуют четкой последовательности выполнения всех перечисленных этапов. В зависимости от сложности решаемой задачи их количество может меняться.

Все приведенные этапы тесно связаны между собой. Например, анализ результатов может привести к необходимости внесения изменений в программу, алгоритм, метод решения или даже в постановку задачи.

2. ПОНЯТИЕ АЛГОРИТМА И ЕГО СВОЙСТВА

Алгоритм – строго определенная конечная последовательность математических и логических действий для решения задачи, обязательно приводящая к некоторому результату.

Простейшими примерами алгоритмов являются правила выполнения четырех основных арифметических действий над числами. Эти правила для десятичной системы еще в IX в. были определены среднеазиатским математиком Аль Хорезми. В результате «европеизации» имени этого ученого и возник термин «алгоритм».

Алгоритм применяется к исходному (входному) набору данных, называемых *аргументами*. Исходные данные задаются до начала работы алгоритма или определяются динамически во время его работы. Число исходных данных может быть равно нулю. В результате выполнения алгоритма получают *результат* – искомый (выходной) набор данных, имеющих определенную алгоритмом связь с входными данными.

Любой алгоритм существует не сам по себе, а предназначен для определенного *исполнителя* (человека, робота, компьютера, языка программирования и т. д.). Совокупность команд, которые конкретный исполнитель умеет выполнять, называется **системой команд исполнителя**.

Алгоритмы характеризуются следующими свойствами.

Дискретность (прерывность, раздельность). Описываемый с помощью алгоритма процесс должен быть разбит на последовательность отдельных шагов, т. е. алгоритм должен состоять из отдельных законченных действий.

Определенность (детерминированность). Это свойство означает, что неоднозначность толкования записи алгоритма недопустима, многократное применение алгоритма к одним и тем же исходным данным должно приводить к одинаковому результату.

Результативность (конечность). Алгоритм обязательно должен приводить к определенному результату за конечное число шагов. Для этого в алгоритме должно быть предусмотрено исключение недопустимых ситуаций (деление на ноль, вычисление логарифма нуля или отрицательного числа и т. п.). Если решения задачи не существует, то в качестве результата может быть выведено сообщение об этом.

Последовательность правил, повлекшая за собой процедуру бесконечного выполнения операций, алгоритмом не является.

Массовость. Это свойство определяет применимость алгоритма ко всем задачам рассматриваемого типа при любых исходных данных.

Формальность. Эта особенность указывает на то, что любой исполнитель, способный воспринимать и выполнять инструкции алгоритма, действует формально, т. е. отвлекается от содержания поставленной задачи, не вникает в ее смысл, а лишь строго выполняет инструкции.

Эффективность. Это свойство, которое позволяет решить задачу за приемлемое время.

Выделяют три крупных класса алгоритмов:

вычислительные алгоритмы выполняют обработку простых структур данных, хотя сам процесс вычисления может быть долгим и сложным;

информационные алгоритмы представляют собой набор процедур, работающих с большими объемами информации (обработка баз данных);

управляющие алгоритмы вырабатывают управляющие воздействия на основе данных внешних процессов (управление технологическими процессами).

3. СПОСОБЫ ПРЕДСТАВЛЕНИЯ АЛГОРИТМА

Алгоритм решения задачи может быть представлен разными способами: с помощью словесного описания, в символах алгоритмического языка, в виде графической схемы или на языке программирования.

Словесное описание (вербальный способ представления алгоритма) отражает структуру алгоритма в произвольной форме с помощью естественного языка с требуемой пошаговой детализацией, однако такая форма алгоритма громоздка, не имеет наглядности, допускает неоднозначность толкования при описании некоторых действий, что делает ее непригодной для реализации в математических расчетах. Примеры:

инструкция по эксплуатации прибора бытовой техники (мультиварки, холодильника и др.), в которой детально описаны режимы его работы;
пошаговый рецепт приготовления кулинарного блюда и т. п.

Алгоритмический язык (символьный способ записи алгоритма) в общем случае представляет собой систему обозначений и правил для единообразной и точной записи алгоритмов и их исполнения. Примером такой системы является *псевдокод*.

Наиболее часто алгоритм решения задачи представляется в виде графической схемы (блок-схемы).

Запись алгоритма на языке программирования называется *программой*.

3.1. Алгоритмический язык (псевдокод)

Псевдокод – описание структуры алгоритма на естественном, частично формализованном языке. Псевдокод не зависит от строго определенного языка программирования, поэтому алгоритм, записанный с его использованием, может быть реализован на любом языке. В псевдокоде применяются формальные конструкции и математические символы.

При записи алгоритма с помощью псевдокода применяются следующие служебные слова и символы, определяющие его базовые конструкции:

АЛГ – АЛГоритм;

НАЧ – НАЧало алгоритма;

КОН – КОНец алгоритма;

если – условие;

то – действия в случае выполнения условия;

иначе – действия в случае невыполнения условия;

все – конец условия;

нц – начало цикла;

кц – конец цикла;

для – цикл с параметром;

пока – условие цикла;

:= – присвоить значение;

// – комментарий.

Логически сгруппированный набор идущих подряд команд псевдокода обычно объединяют в блоки, для выделения которых применяют так называемые

операторные скобки или одинаковые отступы команд от начала строки. В качестве операторных скобок используют фигурные скобки или служебные слова начало и конец.

Последовательность записи алгоритма в псевдокоде:

АЛГ Название алгоритма

НАЧ

Команда 1

Команда 2

.....

КОН

После служебного слова **АЛГ** указывается название алгоритма. Часть алгоритма, заключенная между словами **НАЧ** и **КОН**, называется **телом алгоритма**.

Каждая команда в псевдокоде записывается с новой строки.

Присвоить значения переменным можно следующим образом: $s := 5$, $x := 2 + y$, $m := m - n$. Такого рода запись означает, что сначала выполняется действие над текущими значениями переменных в правой части от знака равенства, а затем полученный результат присваивается переменной, имя которой записано в левой части. Например, в результате операции $k := k + 1$ значение переменной k будет увеличено на единицу.

3.2. Графическая схема алгоритма



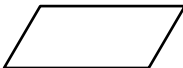
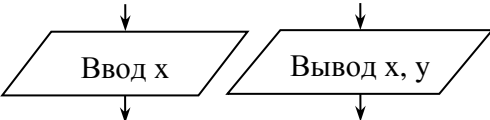

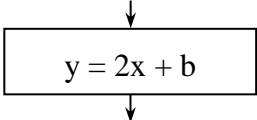
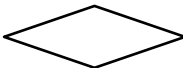
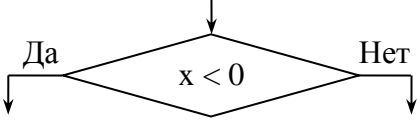
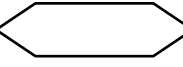
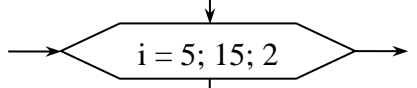



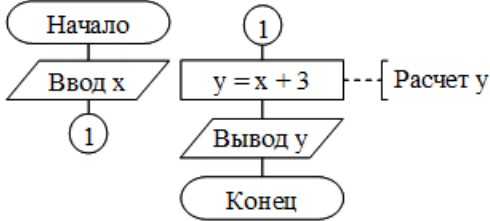


Графическая схема алгоритма (ГСА) – представление алгоритма в виде последовательности функциональных блоков (геометрических фигур) и линий связи (потока) между ними, определяющих порядок выполнения отдельных инструкций.

Конфигурация и размеры блоков в ГСА установлены ГОСТ 19.701-90. Направление линий связи между блоками сверху вниз и слева направо по часовой стрелке считается стандартным, в этом случае линии потока в ГСА можно не сопровождать указанием их направления. В других случаях линии связи обязательно сопровождаются стрелками. Линии связи должны быть направлены к центру блока.

Основные элементы графических схем алгоритмов приведены в [табл. 1](#).

Таблица 1

Основные элементы ГСА

Название блока	Символ в ГСА	Функциональное назначение	Примеры
Пуск, останов		Начало (конец) алгоритма	
Ввод/вывод данных		Ввод информации с помощью устройства ввода или вывод информации на устройство вывода	
Процесс		Вычислительное действие или их последовательность	
Решение (альтернатива, условный блок)		Проверка условия и выбор направления выполнения алгоритма	
Модификация		Цикл с параметром	
Предопределенный процесс		Вычисления по подпрограмме	
Линия потока данных		Связь между блоками	
Соединитель		Разрыв линии потока	
Комментарий		Добавление комментария или пояснительной надписи	

4. БАЗОВЫЕ СТРУКТУРЫ АЛГОРИТМОВ

Любая алгоритмическая конструкция может быть представлена одной из базовых структур алгоритмов или их комбинацией. **Базовая структура алгоритма** – это определенный набор блоков и стандартных способов их соединения для выполнения некоторой типичной последовательности действий. К базовым структурам алгоритмов относятся

- последовательная структура (линейные алгоритмы);
- условная структура (разветвляющиеся алгоритмы);
- циклическая структура (циклические алгоритмы).

Характерной особенностью базовых структур является наличие в них одного входа и одного выхода.

Алгоритмические конструкции могут быть *вложенными*, т. е. могут содержать в себе другие конструкции того же или иного вида.

4.1. Алгоритмы линейной структуры

4.1.1. Понятие линейного алгоритма

Линейный алгоритм – набор команд (инструкций), выполняемых последовательно друг за другом один раз.

В общем виде ГСА линейного процесса представлена на [рис. 1](#).

Линейные алгоритмы являются простейшими алгоритмами, с их помощью может быть реализована только незначительная часть практических задач.

Пример 1. Ввести массу груза в тоннах и последовательно перевести ее в килограммы и граммы.

Обозначим исходную массу, заданную в тоннах, как $m_{\text{т}}$, искомую массу в килограммах – $m_{\text{кг}}$, в граммах – $m_{\text{г}}$. Разные способы записи алгоритма решения задачи представлены в [табл. 2](#).

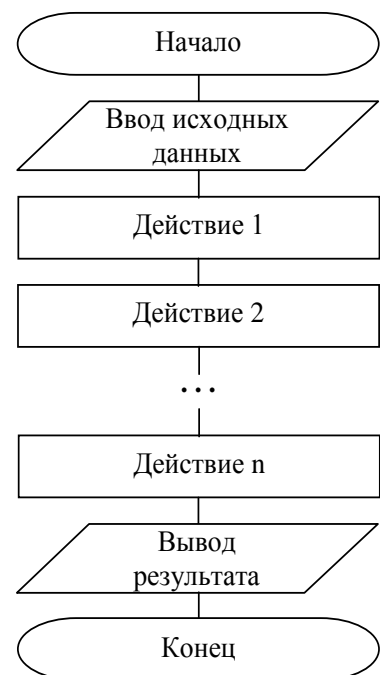


Рис. 1. ГСА линейной структуры

Пример алгоритма линейной структуры

Словесное описание	Псевдокод	ГСА
1. Ввести значение m_T 2. Вычислить $m_{кг} = m_T \cdot 1000$ 3. Вычислить $m_г = m_{кг} \cdot 1000$ 4. Вывести результат	АЛГ Единицы массы НАЧ Ввод m_T Вычисление $m_{кг} := m_T \cdot 1000$ Вычисление $m_г := m_{кг} \cdot 1000$ Вывод $m_T, m_{кг}, m_г$ КОН	<pre> graph TD Start([Начало]) --> Input[/Ввод m_T/] Input --> Calc1[m_кг = m_T * 1000] Calc1 --> Calc2[m_г = m_кг * 1000] Calc2 --> Output[/Вывод m_T, m_кг, m_г/] Output --> End([Конец]) </pre>

4.1.2. Задания

Составить алгоритм в псевдокоде и ГСА для реализации следующей задачи:

В а р и а н т 1. Ввести объем информационного сообщения I в битах и последовательно перевести его в байты, килобайты, мегабайты, обозначив каждую искомую величину отдельной переменной.

В а р и а н т 2. Ввести объем информационного сообщения I в мегабайтах и последовательно перевести его в килобайты, байты, биты, обозначив каждую искомую величину отдельной переменной.

4.2. Алгоритмы разветвляющейся структуры

Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одну проверку условия, в результате которой обеспечивается выбор одного из двух возможных путей (ветвей) решения. После реализации любого выбранного варианта осуществляется переход к общему выходу.

Различают сокращенную (*если-то*) и полную (*если-то-иначе*) формы разветвления.

4.2.1. Сокращенная форма разветвления

Сокращенное разветвление (неполное разветвление, разветвление с обходом) предполагает выполнение заданных действий только по одной ветви (табл. 3). Если проверяемое условие соблюдается (результат – «Да»), то выполняется *Действие 1*. В случае несоблюдения условия (результат – «Нет») разветвление завершает свою работу без выполнения действия.

Таблица 3

Сокращенная форма разветвления

Псевдокод	ГСА
<p>если Условие то Действие 1 // ветвь 1 все // Конец разветвления</p>	

Пример 2. Ввести два числа – t и z . Если t больше z , то вычислить новое значение $t = t + z$. Вывести результирующие числа t и z .

Разные способы записи алгоритма решения задачи представлены в табл. 4.

Таблица 4

Пример алгоритма сокращенной формы разветвления

Словесное описание	Псевдокод	ГСА
1. Ввести числа t и z 2. Проверить выполнение условия $t > z$ 3. Если результат проверки «Да», то вычислить значение $t = t + z$ 4. Вывести t, z	АЛГ Разветвление НАЧ Ввод t, z если $t > z$ то $t := t + z$ все // Конец разветвления Вывод t, z КОН	

4.2.2. Полная форма разветвления

Полное разветвление в зависимости от результата выполнения условия обеспечивает выбор одного из альтернативных путей работы алгоритма, каждый из которых ведет к общему выходу (табл. 5):

- если условие соблюдается (результат – «Да»), то выполняется *Действие 1*;
- если условие не соблюдается (результат – «Нет»), то выполняется *Действие 2* (*Действие 1* не выполняется).

Таблица 5

Полная форма разветвления

Псевдокод	ГСА
<p><u>если</u> Условие <u>то</u> Действие 1 // ветвь 1 <u>иначе</u> Действие 2 // ветвь 2 <u>все</u> // Конец разветвления</p>	<pre> graph TD Start(()) --> Condition{Условие} Condition -- Да --> Action1[Действие 1] Condition -- Нет --> Action2[Действие 2] Action1 --> Join(()) Action2 --> Join Join --> Exit(()) </pre>

В любой форме разветвления действие по каждой ветви (*Действие 1* или *Действие 2*) может содержать в себе не только одну команду, а любую последовательность команд, реализующих базовые структуры алгоритмов, в том числе вложенных одна в другую.

Пример 3. Для произвольного числа x вычислить значение функции y :

$$y = \begin{cases} \pi \cdot \sin x & \text{при } x > 0, \\ \pi \cdot \cos x & \text{в остальных случаях.} \end{cases}$$

Разные способы записи алгоритма решения задачи представлены в табл. 6.

Таблица 6

Пример алгоритма полной формы разветвления

Словесное описание	Псевдокод	ГСА
1. Задать константу π 2. Ввести число x 3. Проверить выполнение условия $x > 0$ 4. Если результат проверки «Да», то вычислить значение $y = \pi \sin x$. В противном случае вычислить значение $y = \pi \cos x$	АЛГ Расчет функции y НАЧ $\pi := 3,14$ Ввод x если $x > 0$ то $y := \pi \cdot \sin x$ // ветвь 1 иначе $y := \pi \cdot \cos x$ // ветвь 2 все // Конец разветвления Вывод x, y КОН	

4.2.3. Задания

Задание 1. В соответствии с индивидуальным вариантом (табл. 7) записать алгоритм в псевдокоде и составить ГСА для расчета значения z . Исходные данные задать следующим образом: a и b – константы, x – произвольное число.

Таблица 7

Индивидуальные варианты заданий

Вариант	Функция	Исходные данные
1	$z = \begin{cases} \operatorname{tg} x + ax^2, & \text{если } x \leq a \cdot b, \\ \sin x^3 + bx & \text{в остальных случаях} \end{cases}$	$a = 2,5; \quad b = -3,7$
2	$z = \begin{cases} ax^3 - b, & \text{если } a \cdot x > b, \\ b \cdot \sin^5 x & \text{в остальных случаях} \end{cases}$	$a = -7; \quad b = 1,4$

Задание 2. В соответствии с индивидуальным вариантом (табл. 8) для фрагментов ГСА, представленных в табл. 9 – 11, выполнить следующие задания:

1) для задания из части А записать алгоритм в псевдокоде и определить результирующие значения переменных при известных исходных данных;

2) для задания из части В вместо вопросительных знаков подобрать значения переменных и операцию отношения ($<$, $>$, $=$, \leq , \geq), приводящие к указанному в ГСА результату, в случаях, если: условие выполняется, условие не выполняется.

Таблица 8

Индивидуальные варианты заданий

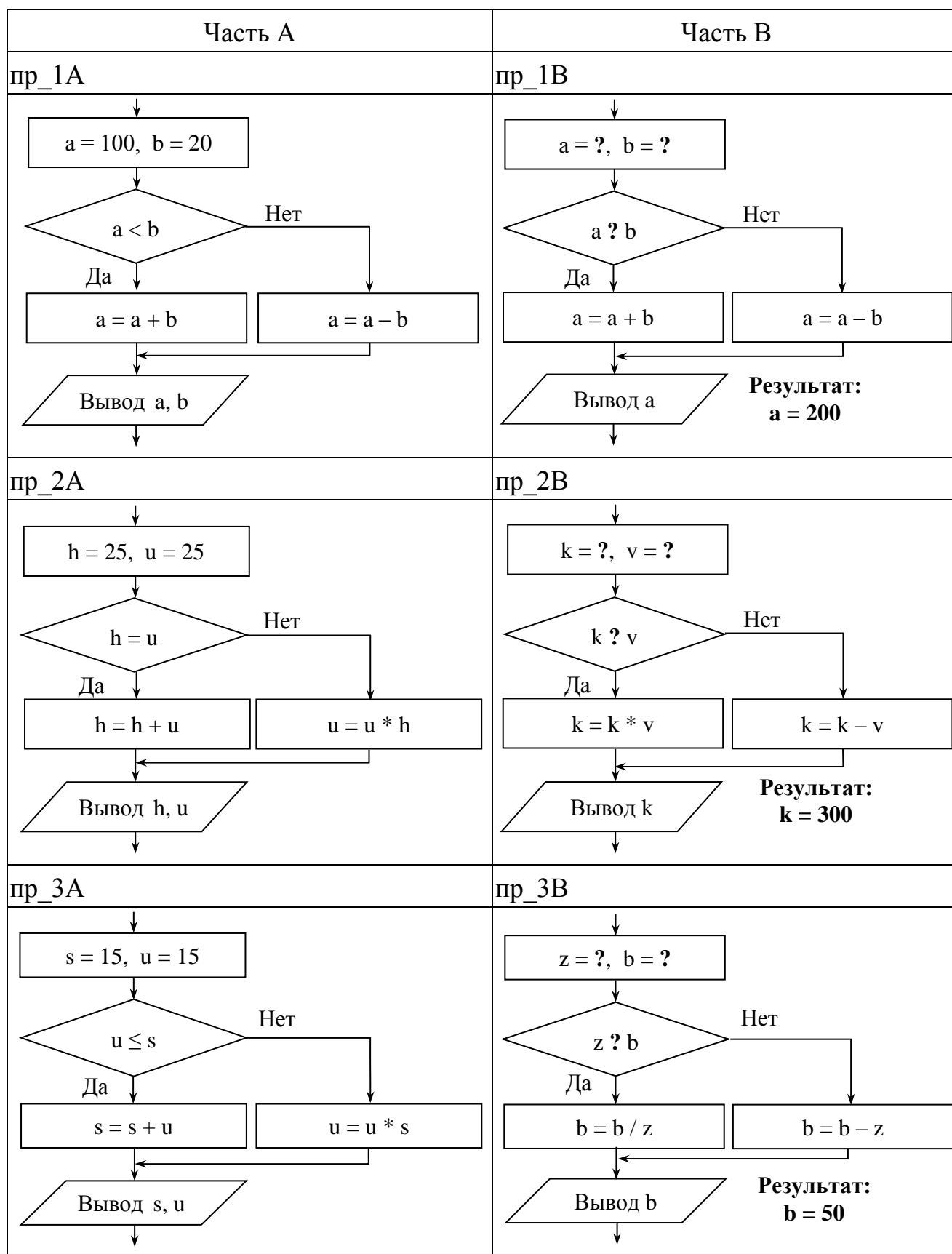
Вариант	Задание из табл. 9	Задание из табл. 10	Задание из табл. 11	
			задание из части А	задание из части В
0	нр_1А	пр_3В	ср_1А	ср_3В
1	нр_2А	пр_2В	ср_2А	ср_2В
2	нр_1В	пр_3А	ср_3А	ср_1В
3	нр_3А	пр_1В	ср_1А	ср_2В
4	нр_2В	пр_1А	ср_2А	ср_1В
5	нр_3В	пр_2А	ср_3А	ср_3В
6	нр_1А	пр_1В	ср_1А	ср_1В
7	нр_1В	пр_2А	ср_2А	ср_3В
8	нр_3А	пр_2В	ср_3А	ср_1В
9	нр_2А	пр_3В	ср_2А	ср_2В
10	нр_2В	пр_3А	ср_3А	ср_1В
11	нр_3В	пр_1А	ср_2А	ср_3В
12	нр_1А	пр_2В	ср_3А	ср_3В
13	нр_1В	пр_1А	ср_1А	ср_2В
14	нр_2В	пр_2А	ср_3А	ср_3В
15	нр_3А	пр_3В	ср_2А	ср_2В
16	нр_2А	пр_2В	ср_1А	ср_1В
17	нр_3В	пр_2А	ср_2А	ср_3В
18	нр_1В	пр_3А	ср_3А	ср_2В

Примечание. В табл. 8 – 11 префиксы в номерах заданий обозначают тематику работы: нр – неполное разветвление; пр – полное разветвление; ср – сложное разветвление.

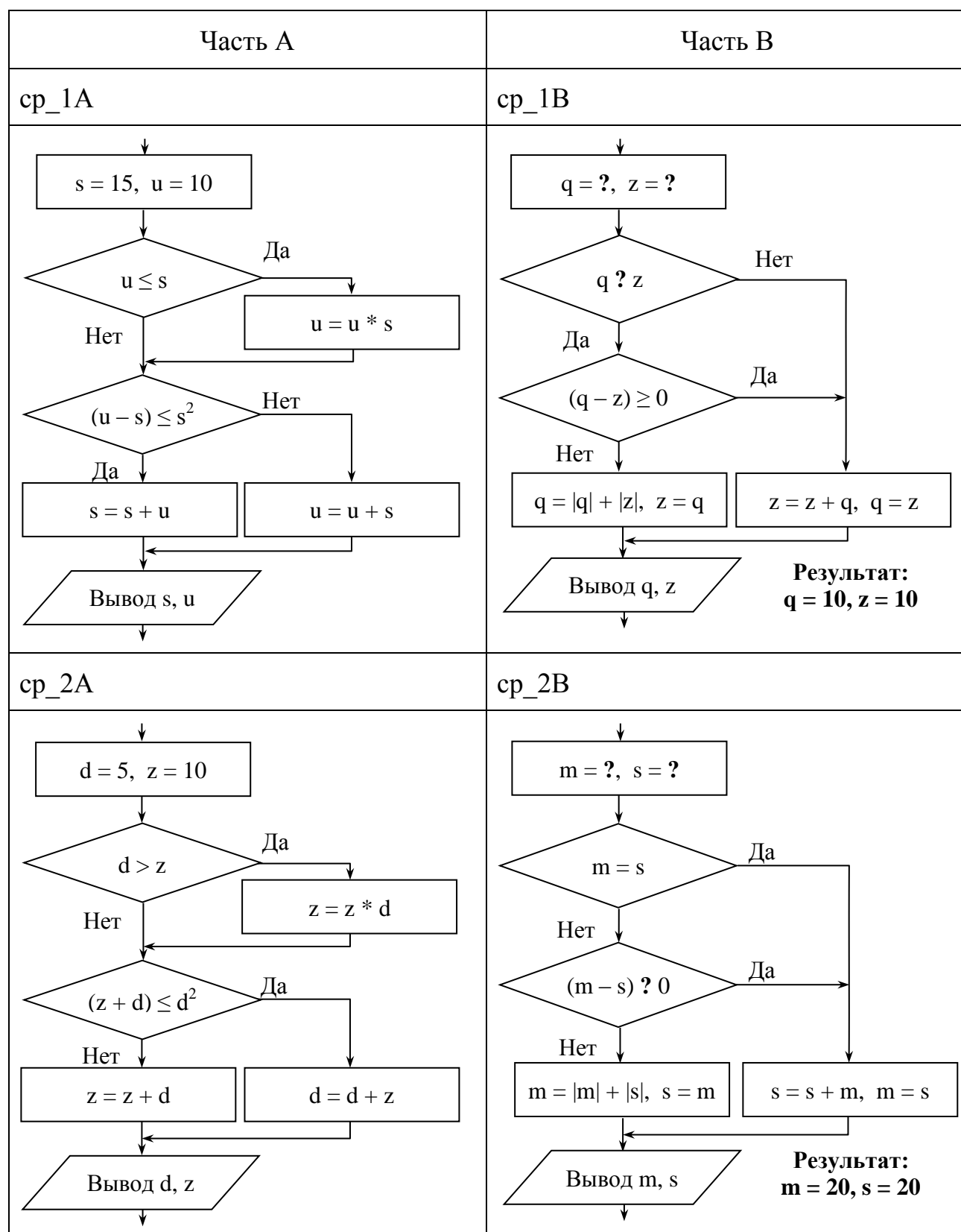
Разветвляющийся алгоритм (сокращенная форма разветвления)

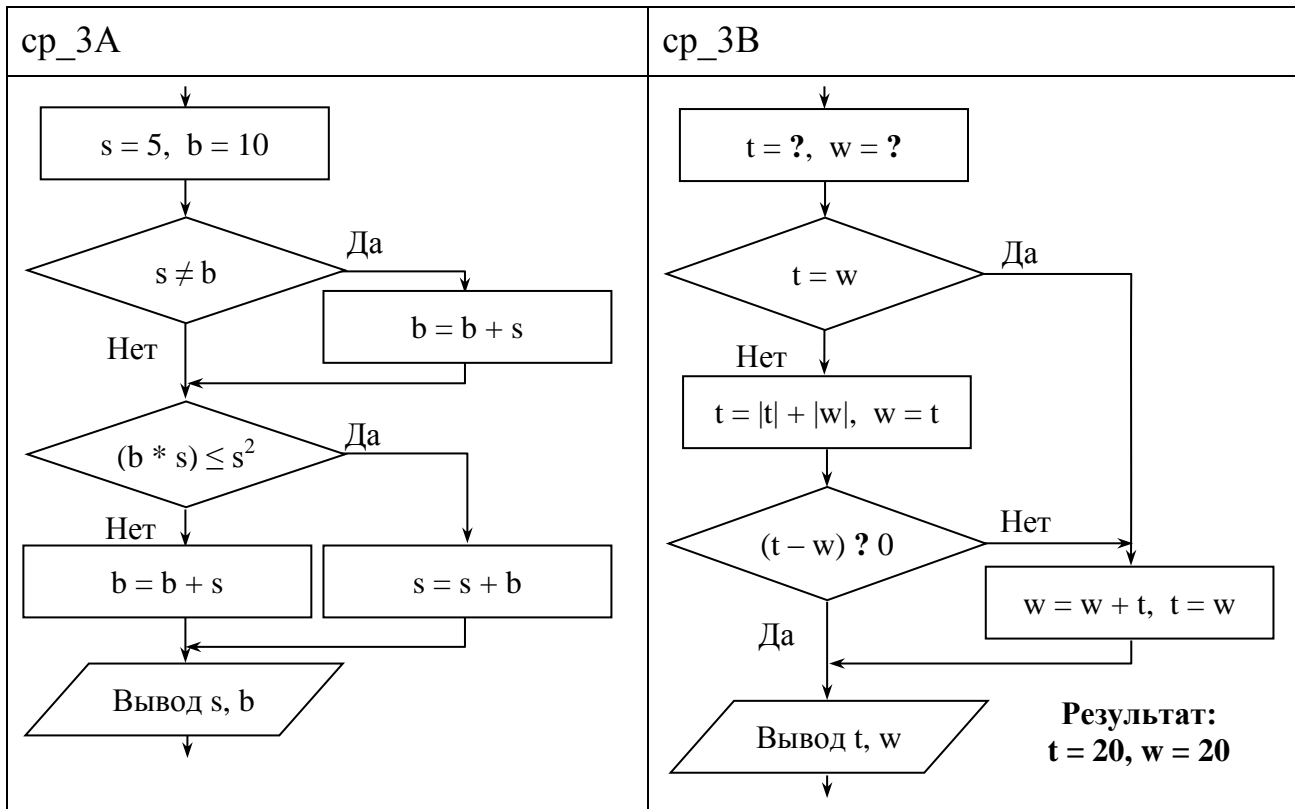
Часть А	Часть В
<p>нр_1А</p> <pre> graph TD Start(()) --> Init["a = 12, b = 22"] Init --> Cond{"a < b"} Cond -- Да --> Assign["b = a"] Assign --> End1(()) Cond -- Нет --> End1 End1 --> Output[/Вывод a, b/] </pre>	<p>нр_1В</p> <pre> graph TD Start(()) --> Init["a = ?, b = ?"] Init --> Cond{"a ? b"} Cond -- Да --> Assign["a = a + b"] Assign --> End2(()) Cond -- Нет --> End2 End2 --> Output[/Вывод a/] Output --> Result["Результат: a = 10"] </pre>
<p>нр_2А</p> <pre> graph TD Start(()) --> Init["n = 28, m = 78"] Init --> Cond{"m > n"} Cond -- Да --> Assign["m = m - n"] Assign --> End3(()) Cond -- Нет --> End3 End3 --> Output[/Вывод n, m/] </pre>	<p>нр_2В</p> <pre> graph TD Start(()) --> Init["k = ?, s = ?"] Init --> Cond{"k ? s"} Cond -- Да --> Assign["k = k / s"] Assign --> End4(()) Cond -- Нет --> End4 End4 --> Output[/Вывод k/] Output --> Result["Результат: k = 10"] </pre>
<p>нр_3А</p> <pre> graph TD Start(()) --> Init["k = 342, f = 42"] Init --> Cond{"f < k"} Cond -- Да --> Assign["f = k - f"] Assign --> End5(()) Cond -- Нет --> End5 End5 --> Output[/Вывод k, f/] </pre>	<p>нр_3В</p> <pre> graph TD Start(()) --> Init["w = ?, s = ?"] Init --> Cond{"w ? s"} Cond -- Да --> Assign["s = s * w"] Assign --> End6(()) Cond -- Нет --> End6 End6 --> Output[/Вывод s/] Output --> Result["Результат: s = 20"] </pre>

Разветвляющийся алгоритм (полная форма разветвления)



Разветвляющийся алгоритм (сложное разветвление)





4.3. Алгоритмы циклической структуры

Цикл – многократное выполнение (повторение) одной и той же последовательности команд, называемой *телом цикла*. Количество повторений (шагов, итераций) тела цикла определяется исходными данными или условием задачи. Любая циклическая конструкция содержит в себе элементы разветвления.

Различают арифметический и итерационный циклы.

4.3.1. Арифметический цикл

Цикл называется **арифметическим**, если количество его повторений заранее известно по условию задачи или может быть легко вычислено (другие названия арифметического цикла – цикл с параметром, с известным числом повторений, регулярный, счетный). Число повторений однозначно определяется законом изменения параметра цикла. Закон изменения параметра отражается в заголовке цикла и характеризуется интервалом (начальным и конечным значениями) параметра и шагом его изменения.

Характерным примером задачи на применение арифметического цикла является расчет значений функции при изменении ее аргумента в определенном диапазоне с заданным шагом. В этом случае аргумент функции является

параметром цикла. Например, при вычислении значений функции $z = \sin t$ для t , изменяющегося в диапазоне $[1; 7]$ с шагом $\Delta t = 2$, параметром цикла будет переменная t , последовательно принимающая четыре значения: $t = 1; 3; 5; 7$, для каждого из которых повторяется расчет z .

Обозначим в общем случае:

x – параметр цикла;

n – начальное значение параметра цикла;

k – конечное значение параметра цикла;

h – шаг изменения значений параметра цикла.

Рассмотрим процесс изменения параметра цикла:

шаг 1 – $x = n$

шаг 2 – $x = n + h$

шаг 3 – $x = n + 2h$

и т. д. . . .

последний шаг – $x = k$

Количество повторений арифметического цикла определяется по формуле:

$$N = \frac{k - n}{h} + 1. \quad (1)$$

Арифметический цикл в псевдокоде записывается следующим образом:

нц

для $x := n; k; h$ // Заголовок цикла

{Тело цикла}

кц

Тело цикла может содержать любую последовательность команд в виде линейной, разветвляющейся, циклической структуры или их различных комбинаций, в том числе вложенных одна в другую.

ГСА арифметического цикла можно изобразить двумя способами: в полной форме – с применением условного блока (рис. 2, а) и в краткой форме – с помощью блока модификации (рис. 2, б).

В полной форме ГСА (см. рис. 2, а) в блоке 1 задается начальное значение аргумента x , в блоке 2 осуществляется сравнение текущего значения x с конечным, затем выполняются команды тела цикла (блок 3). В блоке 4 значение x наращивается, и весь процесс повторяется до тех пор, пока текущее значение x не превысит конечное.

В краткой форме ГСА (см. [рис. 2, б](#)) параметр цикла, его начальное и конечное значения и шаг изменения записываются в блоке модификации 1, который объединяет в себе действия блоков 1, 2 и 4 полной формы ГСА.

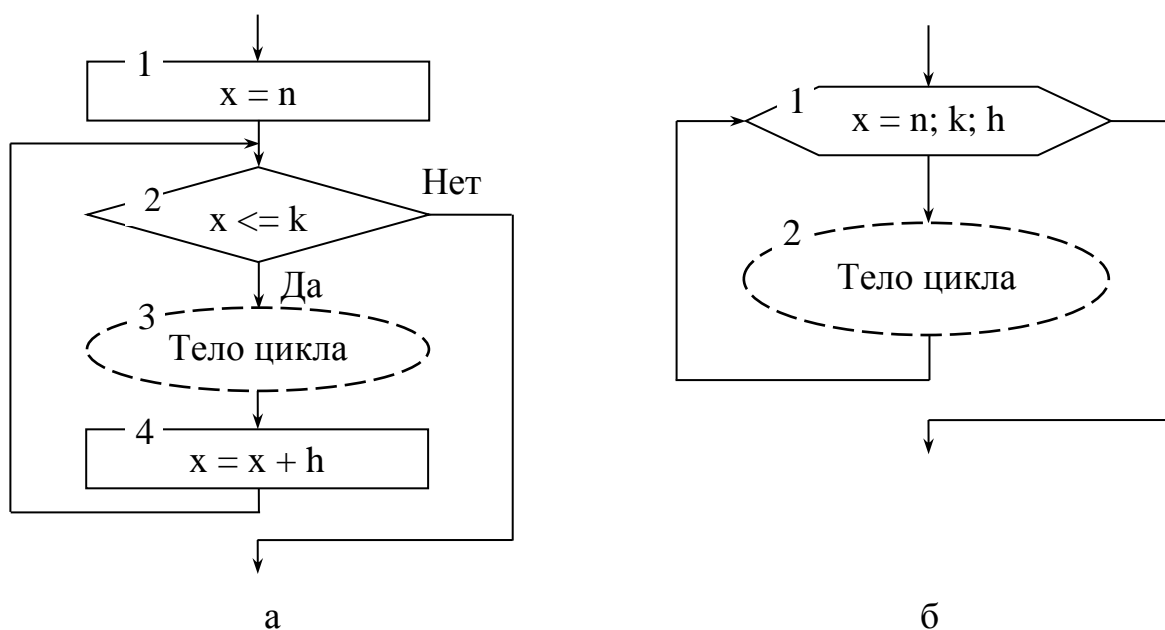


Рис. 2. ГСА арифметического цикла в полной (а) и краткой (б) форме

Пример 4. Вычислить и вывести значения функции $z = a + x^2$, если $a = 2$, значения x изменяются в диапазоне $[3; 7]$ (т. е. $3 \leq x \leq 7$) с шагом $\Delta x = 1$.

Решим эту задачу с помощью арифметического цикла, параметром которого является переменная x . Количество повторений цикла вычисляется по формуле (1): $N = (7 - 3) / 1 + 1 = 5$.

Разные способы записи алгоритма для решения этой задачи представлены в [табл. 12](#), результат его пошагового выполнения, начиная с блока 3, – в [табл. 13](#).

4.3.2. Итерационный цикл

Цикл называется **итерационным**, если число его повторений заранее не известно и определяется некоторым условием. Например, цикл завершается при достижении отрицательного значения указанной переменной. По этой причине ГСА итерационного циклического процесса изображается только в полной форме, использование блока модификации невозможно.

В зависимости от момента проверки условия выполнения циклического процесса итерационные циклы делятся на циклы с предусловием и циклы с постусловием.

Пример алгоритма арифметического цикла

Словесное описание	Псевдокод	ГСА	
		полная	краткая
<p>1. Задать константу a</p> <p>2. Организовать цикл с параметром x: начальное значение – 3, конечное значение – 7, шаг изменения – 1</p> <p>3. Определить тело цикла, которое составляют команды: расчет $z = a + x^2$, вывод x, z</p>	<p>АЛГ Арифметический цикл</p> <p>НАЧ</p> <p>$a := 2$</p> <p>нц</p> <p>для $x := 3; 7; 1$</p> <p>Расчет $z := a + x^2$</p> <p>Вывод x, z</p> <p>кц</p> <p>КОН</p>	<pre> graph TD 1([1 Начало]) --> 2[a = 2] 2 --> 3[x = 3] 3 --> 4{x ≤ 7} 4 -- Да --> 5[Z = a + x²] 5 --> 6[/Вывод x, Z/] 6 --> 7[x = x + 1] 7 --> 4 4 -- Нет --> 8([8 Конец]) </pre>	<pre> graph TD 1([1 Начало]) --> 2[a = 2] 2 --> 3{x = 3; 7; 1} 3 -- Да --> 4[z = a + x²] 4 --> 5[/Вывод x, z/] 5 --> 3 3 -- Нет --> 6([6 Конец]) </pre>

Результат пошагового выполнения арифметического цикла

Номер шага	Значение параметра x	Проверка условия	Результат расчета z	Вывод значений	
				x	z
1	3	$3 \leq 7$ (да)	11	3	11
2	$3 + 1 = 4$	$4 \leq 7$ (да)	18	4	18
3	$4 + 1 = 5$	$5 \leq 7$ (да)	27	5	27
4	$5 + 1 = 6$	$6 \leq 7$ (да)	38	6	38
5	$6 + 1 = 7$	$7 \leq 7$ (да)	51	7	51
6	$7 + 1 = 8$	$8 \leq 7$ (нет)	Выход из цикла		

В *циклах с предусловием* (рис. 3, а) сначала проверяется условие повторения тела цикла, и только в случае его истинности выполняется тело цикла. Цикл выполняется до тех пор, пока условие не станет ложным. Особенность данного цикла заключается в том, что если изначально результат проверки условия отрицателен (*ложь*), то тело цикла не выполнится ни разу. К циклам с предусловием относится также и арифметический цикл (см. п. 4.3.1).

В *циклах с постусловием* (рис. 3, б) сначала выполняются операторы тела цикла, а затем проверяется условие повторения тела цикла. Такие циклы отличаются от циклов с предусловием тем, что тело цикла в них обязательно выполняется хотя бы один раз.

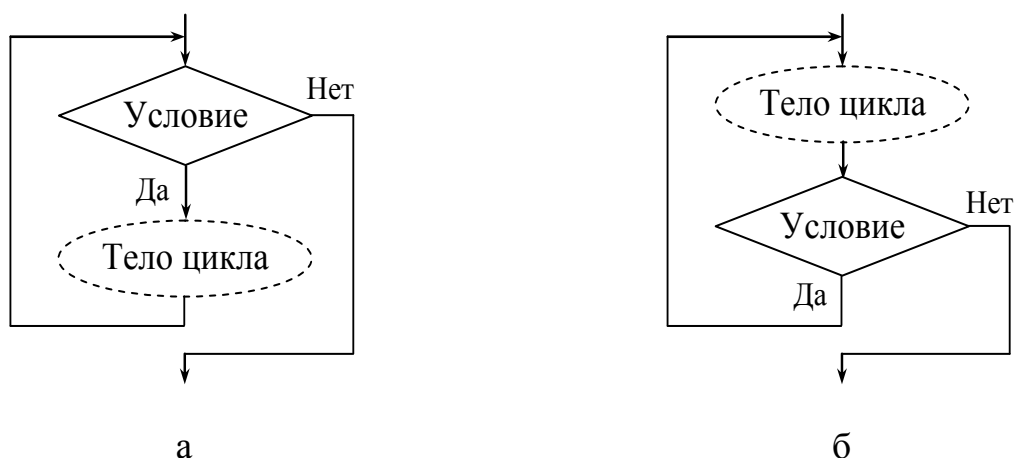


Рис. 3. ГСА итерационного цикла с предусловием (а) и с постусловием (б)

Пример 5. Ввести два произвольных числа – a и b . Пока $b > a$, вычитать из числа b число a .

Количество повторений вычислительного процесса в данном примере зависит от исходных данных и заранее не известно, поэтому для его решения используем итерационный цикл. Переменной цикла является b , тело цикла составляют две команды: расчет и вывод значения этой переменной.

Решим задачу двумя способами – с помощью цикла с предусловием и цикла с постусловием. Соответствующие алгоритмы представлены в табл. 14 и 15.

Таблица 14

Пример алгоритма итерационного цикла с предусловием

Словесное описание	Псевдокод	ГСА
1. Ввести a и b 2. Организовать цикл: проверить условие $b > a$, вычислить $b = b - a$, вывести полученное значение b 3. В случае невыполнения условия п. 2 закончить цикл	АЛГ Цикл с предусловием НАЧ Ввод a, b нц пока $b > a$ $b := b - a$ Вывод b кц КОН	<pre> graph TD Start([Начало]) --> Input[/Ввод a, b/] Input --> Decision{b > a} Decision -- Да --> Process[b = b - a] Process --> Output[/Вывод b/] Output --> Decision Decision -- Нет --> End([Конец]) </pre>

С помощью цикла с предусловием данная задача однозначно реализуется при любых исходных данных. Однако ее корректное решение с помощью цикла с постусловием возможно только в случае, когда исходное значение b больше a . Если это условие нарушено, то применение циклического алгоритма с постусловием будет противоречить условию задачи. Поэтому вид итерационного цикла всегда следует выбирать в зависимости от исходных данных и условия конкретной задачи. В большинстве случаев применяется цикл с предусловием.

Для демонстрации решения примера 5 составим таблицы изменения значений переменных при начальных значениях $a = 3$, $b = 10$: для цикла с предусловием – табл. 16, для цикла с постусловием – табл. 17.

Таблица 15

Пример алгоритма итерационного цикла с постусловием

Словесное описание	Псевдокод	ГСА
1. Ввести a и b 2. Организовать цикл: вычислить $b = b - a$, вывести полученное значение b , проверить условие $b > a$ 3. В случае невыполне- ния условия п. 2 закон- чить цикл	АЛГ Цикл с постусловием НАЧ Ввод a, b нц $b := b - a$ Вывод b пока $b > a$ кц КОН	<pre> graph TD Start([Начало]) --> Input[/Ввод a, b/] Input --> Calc[b = b - a] Calc --> Output[/Вывод b/] Output --> Decision{b > a} Decision -- Да --> Calc Decision -- Нет --> End([Конец]) </pre>

Таблица 16

Значения переменных при реализации цикла с предусловием

Номер итерации	Проверка условия	Значение a	Значение b
—	—	3	10
1	$10 > 3$ (да)	Не изменяется (3)	$10 - 3 = 7$
2	$7 > 3$ (да)	Не изменяется (3)	$7 - 3 = 4$
3	$4 > 3$ (да)	Не изменяется (3)	$4 - 3 = 1$
4	$1 > 3$ (нет)	Выход из цикла	

Таблица 17

Значения переменных при реализации цикла с постусловием

Номер итерации	Значение a	Значение b	Проверка условия
—	3	10	—
1	Не изменяется (3)	$10 - 3 = 7$	$7 > 3$ (да)
2	Не изменяется (3)	$7 - 3 = 4$	$4 > 3$ (да)
3	Не изменяется (3)	$4 - 3 = 1$	$1 > 3$ (нет)
			Выход из цикла

4.3.3. Задания

Задание 1. В соответствии с индивидуальным вариантом (табл. 18) записать алгоритм в псевдокоде и составить ГСА (в полной и краткой формах) для расчета функции z с использованием арифметического цикла. Исходные данные и значение π задать константами.

Таблица 18

Индивидуальные варианты заданий

Вариант	Функция	Исходные данные	Диапазон и шаг изменения аргумента
1	$z = \pi \sin(qt^2)$	$q = 3,2$	$0 \leq t \leq 2,5$ $\Delta t = 0,25$
2	$z = \pi \cos^3(at)$	$a = 2$	$-1 \leq t \leq 2$ $\Delta t = 0,3$

Задание 2. В соответствии с индивидуальным вариантом (табл. 19) для фрагментов ГСА, представленных в табл. 20 – 22, выполнить следующие задания, в каждом из которых обязательно привести таблицу изменения значений переменных:

1) для заданий из части А записать алгоритм в псевдокоде, определить количество повторений цикла и результирующие значения переменных;

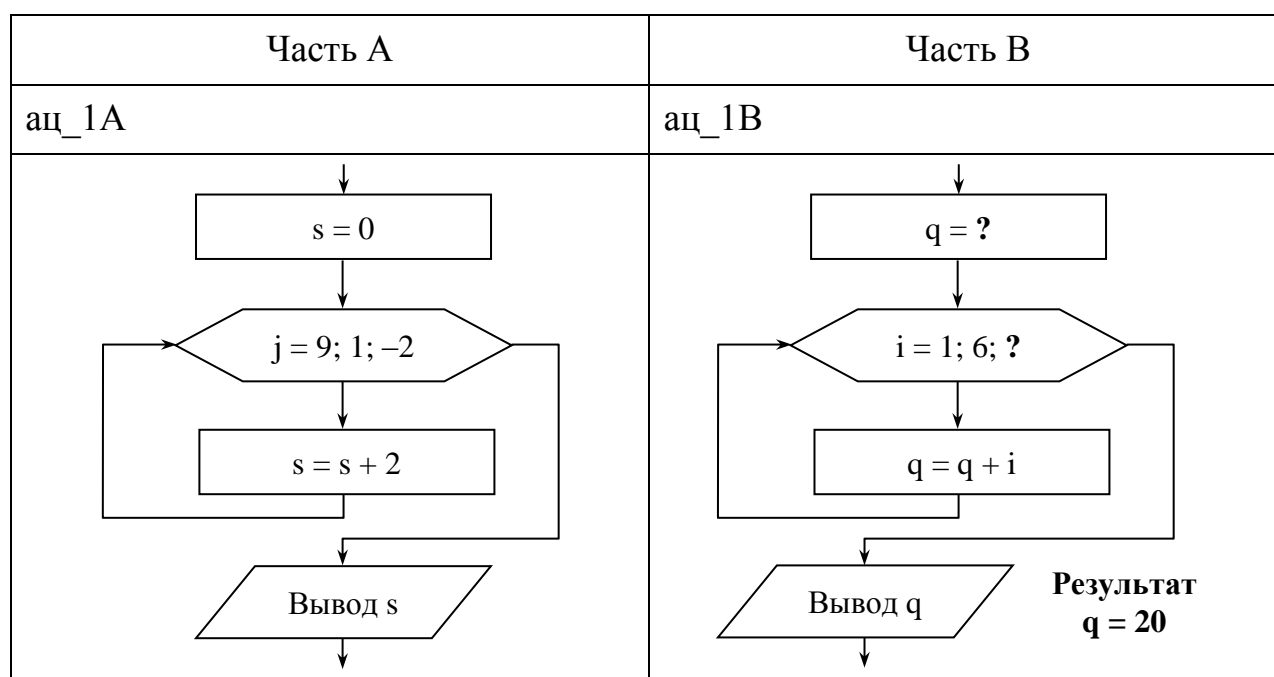
2) для заданий из части В подобрать значения переменных (исходных данных) и (или) операций отношения ($<$, $>$, $=$, \leq , \geq) таким образом, чтобы количество повторений цикла, приводящее к указанному в ГСА результату, было не менее трех.

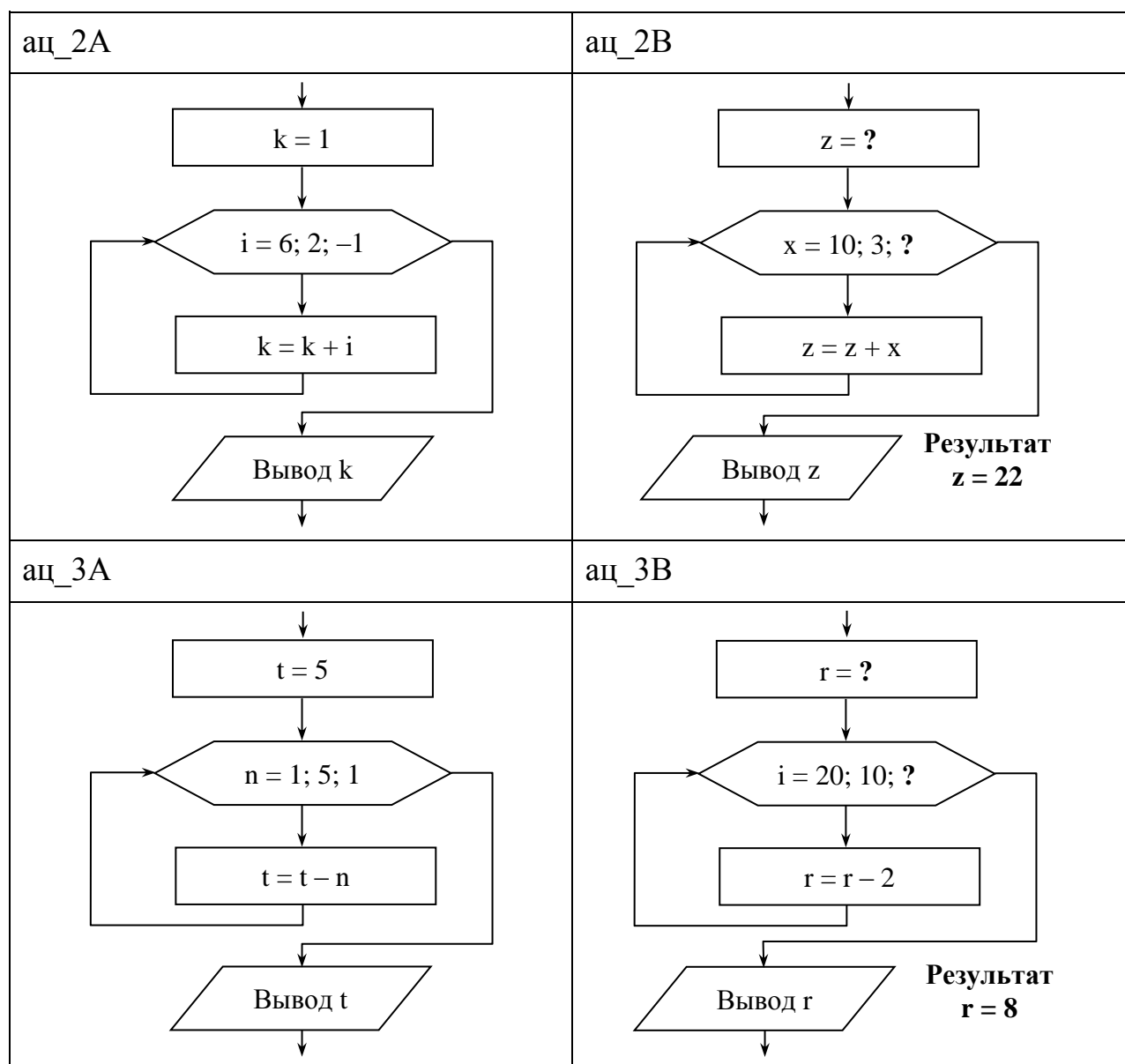
Индивидуальные варианты заданий

Вариант	Задание из табл. 20	Задание из табл. 21	Задание из табл. 22	Вариант	Задание из табл. 20	Задание из табл. 21	Задание из табл. 22
0	ац_1А	ицн_3А	ицк_1В	10	ац_2В	ицн_3В	ицк_2А
1	ац_2А	ицн_2А	ицк_2В	11	ац_3В	ицн_1В	ицк_1А
2	ац_1В	ицн_3В	ицк_3А	12	ац_1А	ицн_2А	ицк_3В
3	ац_3А	ицн_1А	ицк_3В	13	ац_1В	ицн_1А	ицк_1В
4	ац_2В	ицн_1В	ицк_2А	14	ац_2В	ицн_2А	ицк_2В
5	ац_3В	ицн_2В	ицк_1А	15	ац_3А	ицн_3В	ицк_1А
6	ац_1А	ицн_1А	ицк_2В	16	ац_2А	ицн_2В	ицк_3А
7	ац_1В	ицн_2В	ицк_3А	17	ац_3В	ицн_2А	ицк_2А
8	ац_3А	ицн_2А	ицк_1В	18	ац_1В	ицн_3В	ицк_3А
9	ац_2А	ицн_3А	ицк_3В				

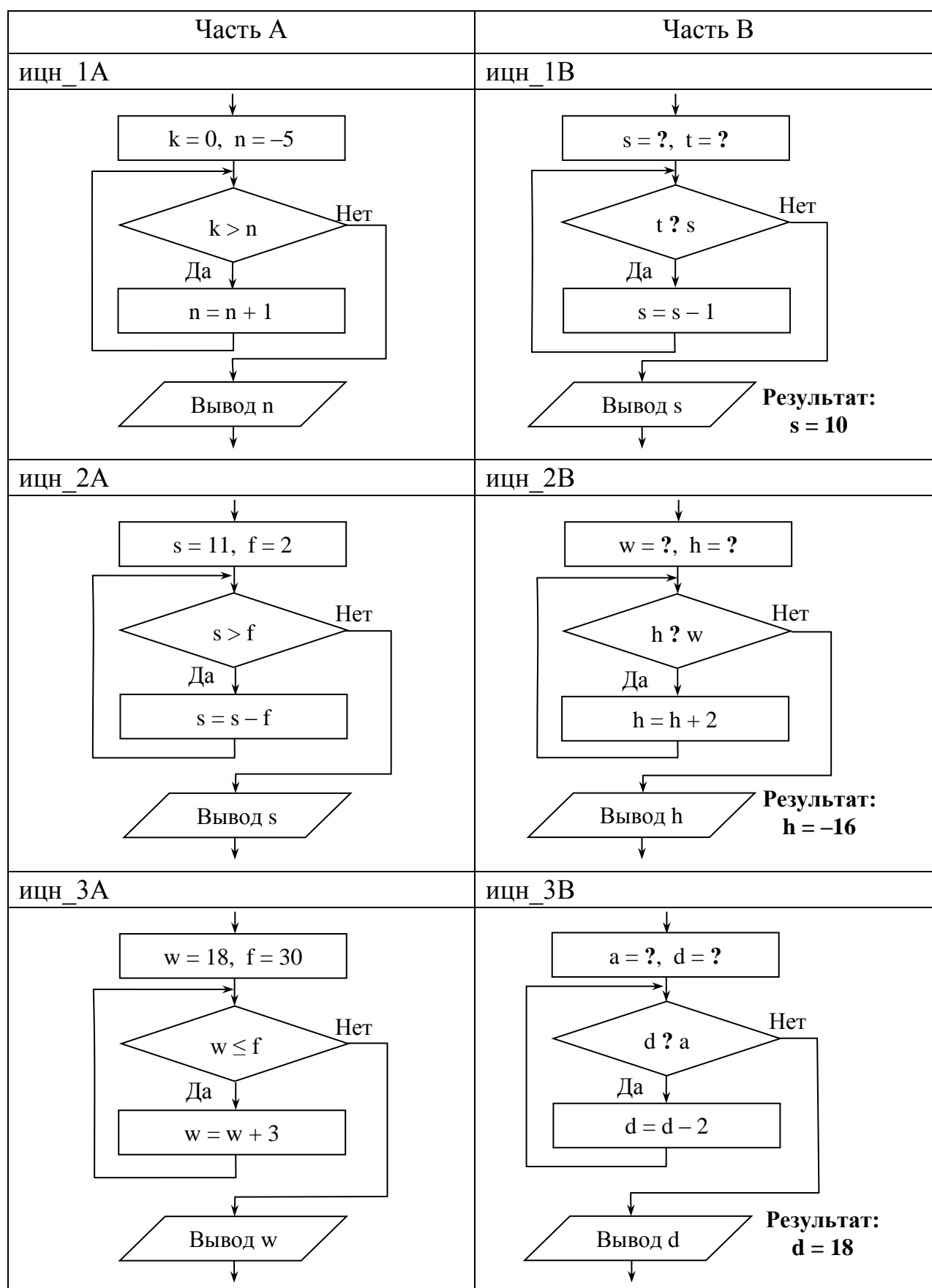
Примечание. В табл. 19 – 22 префиксы в номерах заданий обозначают тематику работы: ац – арифметический цикл; ицн – итерационный цикл с предусловием (т. е. с проверкой условия в начале цикла); ицк – итерационный цикл с постусловием (т. е. с проверкой условия в конце цикла).

Арифметический цикл

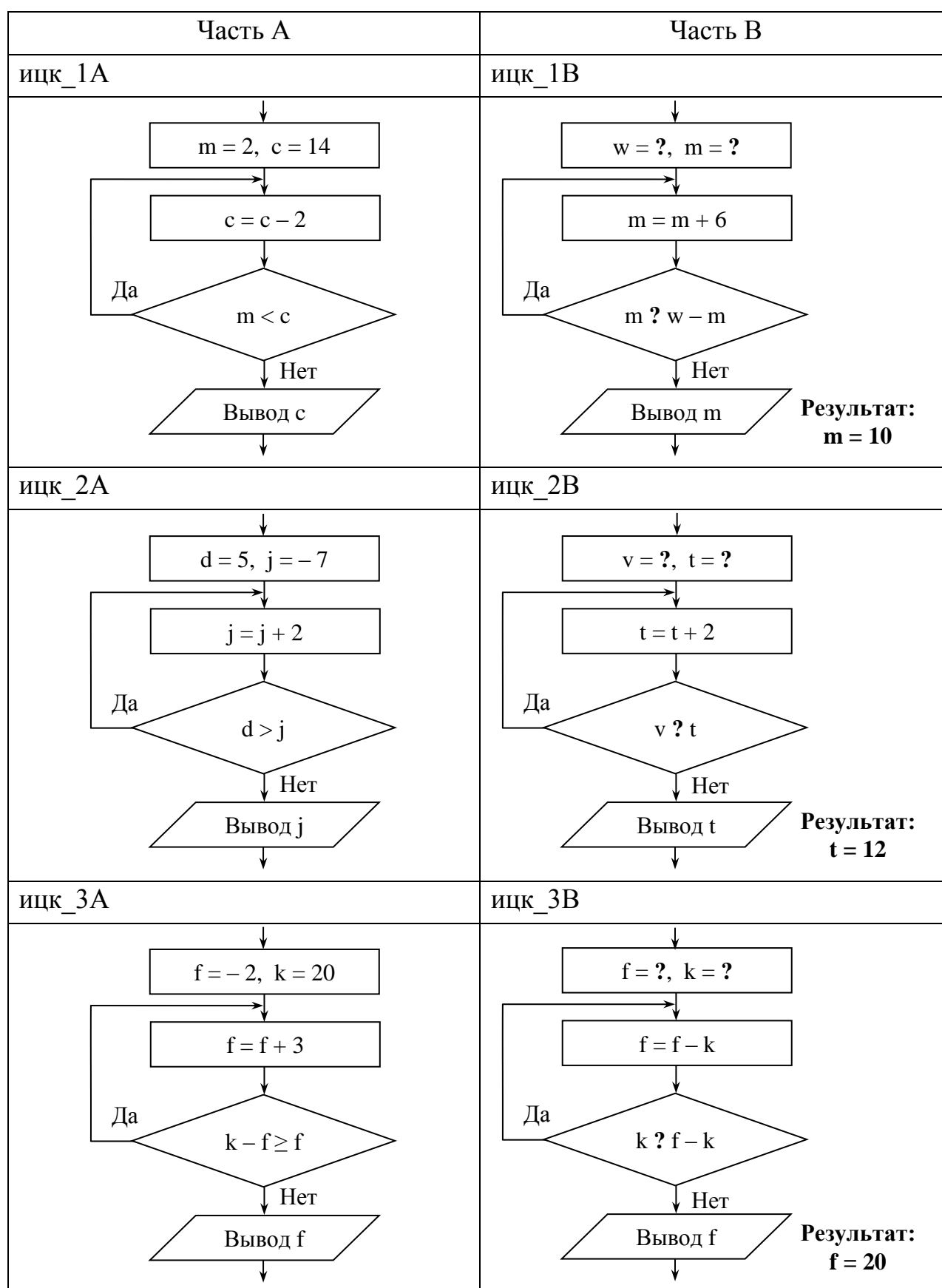




Итерационный цикл с предусловием



Итерационный цикл с постусловием



5. КОНТРОЛЬНЫЕ ВОПРОСЫ

- 1) Опишите этапы решения задач на компьютере.
- 2) Что называется алгоритмом?
- 3) Перечислите свойства алгоритмов.
- 4) Назовите существующие способы представления алгоритмов.
- 5) Перечислите базовые структуры алгоритмов.
- 6) Что такое ГСА?
- 7) Чем отличается полная форма разветвления от сокращенной?
- 8) Какой цикл называется арифметическим?
- 9) Чем отличается полная форма ГСА арифметического цикла от краткой?
- 10) Какой цикл называется итерационным?
- 11) В чем отличие цикла с предусловием от цикла с постусловием?

6. ПРИМЕРЫ ТЕСТОВЫХ ВОПРОСОВ

Вопрос № 1 (один верный ответ)

Результатом выполнения представленного ниже фрагмента алгоритма является ...

$N := 0$

НЦ

для $k := 1; 5$

 ввод x

$N := N + x$

кц

$N := N / 5$

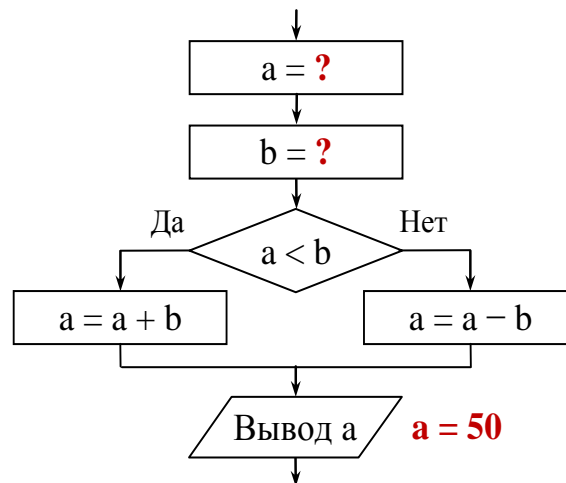
 вывод N

Варианты ответов:

- 1) остаток от деления числа N на 5;
- 2) среднее арифметическое пяти чисел, введенных с клавиатуры;
- 3) сумма пяти чисел, введенных с клавиатуры.

Вопрос № 2 (несколько верных ответов)

Подобрать исходные значения переменных a и b так, чтобы после выполнения алгоритма получился указанный справа от блока вывода результат.

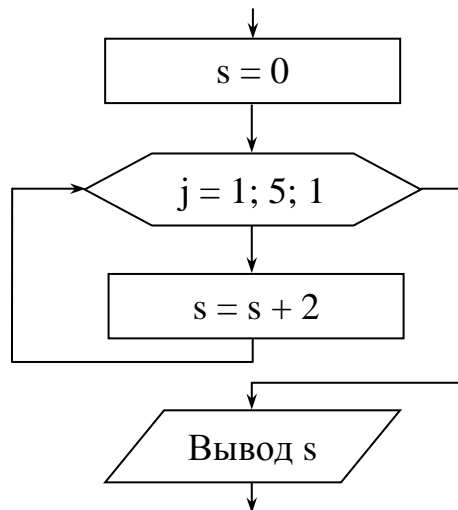


Варианты ответов:

- 1) $a = 20, b = 30$;
- 2) $a = 50, b = 100$;
- 3) $a = 70, b = 20$;
- 4) $a = 50, b = 50$.

Вопрос № 3 (один верный ответ)

Что является параметром цикла в ГСА, представленной на рисунке?

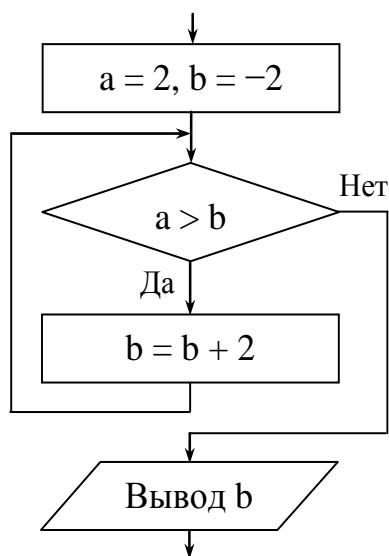


Варианты ответов:

- 1) переменная j ;
- 2) переменная s ;
- 3) выражение $s + 2$.

Вопрос № 4 (один верный ответ)

Определите значение результирующей переменной после выполнения представленного на рисунке фрагмента ГСА.



Варианты ответов:

- 1) 2;
- 2) 0;
- 3) -2;
- 4) нет правильного ответа.

Библиографический список

1. Трофимов В. В. Алгоритмизация и программирование: Учебник / В. В. Трофимов, Т. А. Павловская. М.: Юрайт, 2019. 137 с.
2. Черпаков И. В. Основы программирования: Учебник и практикум для прикладного бакалавриата / И. В. Черпаков. М.: Юрайт, 2019. 219 с.
3. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. М.: Изд-во стандартов, 1990. 36 с.

Учебное издание

СИДОРОВА Елена Анатольевна, ЖЕЛЕЗНЯК Светлана Петровна,
МАНОХИНА Татьяна Витальевна, СТУПАКОВ Сергей Анатольевич

ОСНОВЫ АЛГОРИТМИЗАЦИИ

Учебно-методическое пособие

Редактор Н. А. Майорова

Подписано в печать .02.2020. Формат $60 \times 84 \frac{1}{16}$.
Офсетная печать. Бумага офсетная. Усл. печ. л. 2,3. Уч.-изд. л. 2,5.
Тираж 100 экз. Заказ .

**

Редакционно-издательский отдел ОмГУПСа
Типография ОмГУПСа

*

644046, г. Омск, пр. Маркса, 35