

Министерство образования и науки Российской Федерации  
Государственное образовательное учреждение  
высшего профессионального образования  
«Тихоокеанский государственный университет»

**Т. А. Бочарова**  
**Н. О. Бегункова**

## **ОСНОВЫ АЛГОРИТМИЗАЦИИ**

*Утверждено*  
*издательско-библиотечным советом университета*  
*в качестве учебного пособия*

Хабаровск  
Издательство ТОГУ  
2011

УДК 004 (075.8)  
ББК 3 973.2 – 018  
Б 865

Рецензенты:

Кафедра «Информатика и информационные процессы» ДВГГУ  
(заведующий кафедрой кандидат физико-математических наук, доцент *В. А. Казинец*)  
Кандидат педагогических наук, доцент *Н. П. Табачук*

Научный редактор доктор технических наук *С. П. Исаев*

**Бочарова Т. А.**

Б 865 Основы алгоритмизации : учеб. пособие / Т. А. Бочарова, Н. О. Бегункова. – Хабаровск : Изд-во Тихоокеан. гос. ун-та, 2011. – 64 с.  
ISBN 978-5-7389-0966-5

Алгоритмизация – это раздел информатики, изучающий базовые конструкции и принципы программирования. В пособии рассмотрены понятие алгоритма, способы описания алгоритмов, основные свойства и характеристики алгоритмов, алгоритмические структуры и правила их оформления. Излагаются основные принципы и типовые приемы алгоритмизации, приводятся примеры алгоритмов решения типовых задач. Пособие направлено на формирование начальных навыков алгоритмизации, составляющих фундаментальную базу при изучении основ программирования. Особенностью данного пособия является то, что материал адаптирован к уровню подготовки студентов-первокурсников.

Предназначено для студентов всех специальностей, изучающих дисциплину «Информатика», преподавателей высших и средних учебных заведений, а также для пользователей, которые делают первые шаги в программировании.

УДК 004 (075.8)  
ББК 3 973.2 – 018

ISBN 978-5-7389-0966-5

© Тихоокеанский государственный университет, 2011  
© Бочарова Т. А., 2011  
© Бегункова Н. О., 2011

## **ВВЕДЕНИЕ**

Любой человек ежедневно встречается с множеством повседневных и профессиональных задач. Для решения многих из них существуют определенные правила (инструкции, предписания), объясняющие, как решать определенную задачу. В процессе решения можно применять готовые правила или формулировать собственные. Чем точнее и понятнее описаны правила решения задач, тем быстрее человек овладеет ими и будет эффективнее их применять. Решение многих задач человек передает техническим устройствам – ПК, автоматам, роботам и т. д. Их применение предъявляет очень строгие требования к точности описания правил и последовательности выполнения действий. Поэтому разрабатываются специальные алгоритмы для четкого и строгого описания различных правил.

Алгоритмизация – это раздел информатики, изучающий методы и приемы построения алгоритмов, а также их свойства. Она является основным, базовым компонентом компьютерной грамотности в современном компьютерном мире. Для достижения положительных результатов важную роль играет умение разрабатывать оптимальный алгоритм решения поставленной задачи, что требует от исполнителя наличия определенных навыков алгоритмизации и системного анализа, а также знания математики, физики, химии, экономики и других дисциплин.

Основу деятельности специалиста практически любой области составляет умение ставить задачи, разрабатывать алгоритмы, получать решения, производить анализ полученных данных и делать выводы. Поэтому в своей будущей профессиональной деятельности студенты должны уметь грамотно применять персональный компьютер (ПК) для решения научных и производственных задач.

В данном пособии рассмотрены основные структуры алгоритмов и типовые приемы алгоритмизации. Приведены примеры и задания, охватывающие основные типы вычислительных процессов.

## ОСНОВНЫЕ ЭТАПЫ РЕШЕНИЯ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ КОМПЬЮТЕРА

Создание компьютерной программы – очень сложный, многоэтапный и трудоемкий процесс. Он включает в себя последовательность действий от постановки задачи до получения решения.

**1. Общая формулировка задачи.** На этом этапе задача формулируется в содержательных терминах, определяются входные и выходные данные задачи.

**2. Математическая формулировка задачи.** На этом этапе определяются математические величины, которые будут описывать задачу, а также математические связи между ними, т. е. составляется математическая модель. Неправильная или плохая модель обречет на неудачу весь дальнейший проект, поэтому этот этап является крайне важным.

**3. Выбор метода решения.** Исходя как из субъективных причин (знание тех или иных математических методов), так и объективных (имеющиеся ресурсы), из большого количества математических методов выбирается тот, который целесообразно использовать для решения поставленной задачи.

**4. Составление алгоритма решения.** На этом этапе должна четко прослеживаться связь с предыдущим. В ходе него разрабатывается эффективный алгоритм, т. е. такой, реализация которого потребует наименьшего количества ресурсов компьютера.

**5. Составление и отладка программы.** На этом этапе применяются основные правила записи и преобразования команд, записанных на естественном языке, на язык машинных кодов.

**6. Тестирование программы.** Происходит подтверждение или опровержение правильности работы алгоритма. Для этого, как правило, решаются задачи с такими исходными данными, для которых известно достоверное решение, либо применяются косвенные свидетельства.

**7. Решение поставленной задачи и представление результатов.** На данном этапе осуществляется удобный и наглядный вывод результатов.

При решении конкретных задач некоторые из этих этапов могут исключаться самой постановкой задачи. Для каждого из этапов создания и использования программы существуют определенные приемы обеспечения

качества программы. Большую роль в создании продуктов высокого качества играет глубина и тщательность проработки схемы алгоритма.

На этапе разработки алгоритма рекомендуется придерживаться следующих правил его составления:

1. Алгоритм должен быть максимально прост и понятен.
2. Алгоритм должен состоять из мелких шагов.
3. Сложная задача должна разбиваться на достаточно простые, легко воспринимаемые части (блоки).
4. Логика алгоритма должна опираться на минимальное число достаточно простых базовых управляющих структур.

В итоге процесс разработки алгоритма должен быть направлен на получение четкой структуры алгоритмических конструкций.

## ПОНЯТИЕ АЛГОРИТМА

Понятие алгоритма относится к числу основных понятий современной вычислительной математики и информатики.

*Алгоритм* — описанная на некотором языке точная конечная система правил, определяющая содержание и порядок действий над некоторыми объектами, строгое выполнение которых дает решение поставленной задачи.

Или (более коротко):

*Алгоритм* — это строго определенная последовательность действий, необходимых для решения данной задачи.

Разработать алгоритм решения означает разбить задачу на последовательно выполняемые этапы. Можно сказать, что алгоритм описывает процесс преобразования исходных данных в результаты, т. к. для решения любой задачи необходимо:

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

Алгоритм предназначен всегда для определенного *исполнителя* – человека, робота, компьютера, языка программирования и т. д. Умение выполнять

определенные команды является *свойством*, характеризующим любого исполнителя. Совокупность всех команд, которые данный исполнитель может выполнять, называется *системой команд* исполнителя. Соответственно алгоритм описывается в командах определенного исполнителя, который будет его реализовывать. Объекты, над которыми исполнитель может совершать действия, образуют *среду исполнителя*.

Разработка алгоритма решения задачи – это разбиение задачи на последовательно выполняемые этапы. Результаты выполнения предыдущих этапов могут использоваться при выполнении последующих. Содержание каждого этапа и порядок выполнения этапов должны быть четко указаны. Отдельный этап алгоритма должен быть простым и понятным без пояснений, либо представлять собой другую, более простую задачу, алгоритм решения которой известен (разработан заранее). Поэтому описание алгоритма решения задачи выполняется в соответствии со следующими правилами:

Определяются исходные данные задачи.

2. Процесс решения задачи разбивается на этапы, понятные и однозначные для исполнителя.
3. Указывается порядок, в котором выполняются этапы, а также признак завершения процесса.
4. Определяется, что является результатом решения задачи.

## **СВОЙСТВА АЛГОРИТМА**

Значение слова «алгоритм» является синонимом таких понятий, как «набор инструкций», «последовательность действий», «метод». Однако, в отличие от них, алгоритм характеризуется определенными свойствами. Свойства алгоритма – это набор свойств, отличающих алгоритм от любых предписаний и обеспечивающих его автоматическое исполнение. Алгоритм обладает следующим набором основных свойств: дискретностью, массовостью, формальностью, результативностью, определенностью.

*Дискретность* (разрывность) – это свойство алгоритма, характеризующее его структуру: каждый алгоритм состоит из отдельных законченных действий, т. е. «делится на шаги».

*Массовость* – применимость алгоритма ко всем задачам рассматриваемого типа, при любых исходных данных.

*Определенность* (детерминированность, точность) – свойство алгоритма, указывающее на то, что каждый шаг алгоритма должен быть строго определен и не допускать различных толкований; также строго должен быть определен порядок выполнения отдельных шагов.

*Результативность* – свойство, состоящее в том, что любой алгоритм должен завершаться за конечное (пусть даже очень большое) число шагов.

*Формальность* – это свойство указывает на то, что любой исполнитель, способный воспринимать и выполнять инструкции алгоритма, действует формально, т. е. отвлекается от содержания поставленной задачи и лишь строго выполняет инструкции. Другими словами, механически выполняя все указанные в алгоритме этапы в требуемом порядке, исполнитель может всегда правильно решить задачу.

Процесс разработки алгоритма называется **алгоритмизацией** и требует четкого и полного понимания задачи. Сущность алгоритмизации вычислительного процесса проявляется в следующих действиях, отражающих его свойства:

- выделении законченных частей вычислительного процесса;
- формальной записи каждого из них;
- назначении определенного порядка выполнения выделенных частей;
- проверки правильности выбранного алгоритма по реализации заданного метода вычислений.

## **СПОСОБЫ ОПИСАНИЯ АЛГОРИТМОВ**

На любой стадии существования алгоритмы представляют с помощью конкретных изобразительных средств, состав и правила употребления которых образуют конкретные способы или формы записи. К настоящему времени сложились пять наиболее употребительных способов записи:

словесное описание;

формульно-словесное описание;

псевдокод;

графический способ (блок-схема);

программа (способ описания с помощью языков программирования).

## **Словесное описание**

Словесное описание представляет алгоритм – инструкцию о выполнении действий в определенной последовательности с помощью слов и предложений естественного языка. Форма изложения произвольна и устанавливается разработчиком.

Этот способ описания не имеет широкого распространения, т. к. строго не формализуем (под «формальным» понимается то, что описание абсолютно полное и учитывает все возможные ситуации, которые могут возникнуть в ходе решения), допускает неоднозначность толкования при описании некоторых действий, страдает многословностью.

### **Пример 1**

Алгоритм нахождения наибольшего общего делителя (НОД) двух натуральных чисел.

- 1) задать два числа;
- 2) если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма;
- 3) определить большее из чисел;
- 4) заменить большее из чисел разностью большего и меньшего из чисел;
- 5) повторить алгоритм с п. 2.

### **Пример 2**

Алгоритм выполнения домашнего задания по математике.

- 1) открыть дневник;
- 2) посмотреть, что задано;
- 3) взять учебник и тетрадь по предмету;
- 4) прочитать параграф и выполнить письменные упражнения;
- 5) сравнить полученные результаты с ответами в конце учебника;
- 6) если ответы совпадают, закрыть тетрадь и учебник, если нет – повторить алгоритм с п. 4.



## Формульно-словесный способ

Формульно-словесный способ записи действий содержит формальные символы и выражения (формулы) в сочетании со словесными пояснениями. Т. е. алгоритм записывается в виде текста с формулами по пунктам, определяющим последовательность действий. Этот способ описания нагляден, лаконичен, но не является строго формальным.

### Пример 3

Алгоритм вычисления следующего выражения:  $y = 2a - (x + 6)$ .

- 1) ввести значения  $a$  и  $x$ ;
- 2) найти сумму  $(x + 6)$ ;
- 3) найти произведение  $(2 * a)$ ;
- 4) вычислить  $y$  как разность  $y = 2a - (x + 6)$ ;
- 5) вывести  $y$  как результат вычисления выражения.

### Пример 4

Алгоритм решения задачи по геометрии.

- 1) Дано: высота треугольника  $AH=2$  см, основание треугольника  $BC=5$  см.
- 2) Найти: площадь  $S_{\triangle ABC}$ .
- 3) Решение: Площадь треугольника находится по формуле  $S_{\triangle} = \frac{1}{2} AH * BC$ .
- 4) Подставим данные задачи:  $S_{\triangle} = \frac{1}{2} 2 * 5$ .
- 5) Результат:  $S_{\triangle ABC} = 5$  см.

## Псевдокод

Псевдокод представляет собой описание структуры алгоритма на естественном, частично-формализованном языке, позволяющее выявить основные этапы решения задачи перед точной его записью на языке программирования.

В псевдокоде используются некоторые формальные конструкции и общепринятая математическая символика. Данный способ тесно связан со структурным подходом к программированию. Псевдокод занимает промежуточное положение между естественным языком и языком

программирования. Его применяют преимущественно для того, чтобы подробнее объяснить работу программы, что облегчает проверку правильности программы.

#### Пример 5

Алгоритм сложения двух чисел.

**Псевдокод:**

- 1) Ввод двух чисел  $a$  и  $b$ .
- 2) Вычисление суммы  $S = a + b$ .
- 3) Вывод  $S$ .
- 4) Конец.

#### Пример 6

Алгоритм заполнения зачетной ведомости группы из 20 студентов ( $i$  – номер студента).

**Псевдокод**

**начало цикла**

**для**

$i$  от 1 до 20 с шагом 1

**повторять:**

1.1. ввести фамилию студента

1.2. поставить оценку.

**конец цикла (кц)**

вывод ведомости

## Графическая запись

Графическая запись, или **блок-схема**, – описание структуры алгоритма с помощью геометрических фигур с линиями связями, показывающими порядок выполнения отдельных инструкций. Описание алгоритмов с помощью схем – один из наиболее наглядных и компактных способов. Этот способ имеет ряд преимуществ перед остальными:

- наглядное отображение базовых конструкций алгоритма;
- концентрация внимания на структуре алгоритма;

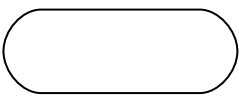
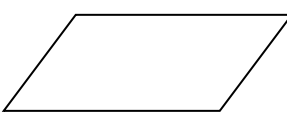

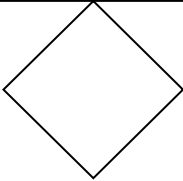
- использование принципа блочности при коллективном решении сложной задачи;
- преобразование алгоритма методом укрупнения (сведения к единому блоку) или детализации (разбиения на ряд блоков);
- быстрая проверка разработанного алгоритма.

В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т. п.) соответствует геометрическая фигура, называемая блоком. Блочные символы соединяются линиями переходов (стрелками), определяющими очередность выполнения действий.



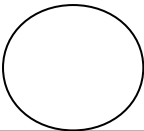
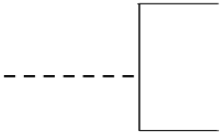
Условные графические изображения, используемые при построении схем, называются символами. Система символов и правила построения алгоритмов определены соответствующими стандартами: блок-схема выстраивается в одном направлении: либо сверху вниз, либо слева направо, в порядке выполнения действий.

Для наглядности операции разного вида изображаются в схеме различными геометрическими фигурами (таблица).

### Средства графического изображения алгоритмов

Наименование	Символ	Выполняемые действия
Терминатор (пуск-остановка)		Обозначает начало и конец алгоритма
Данные (ввод-вывод)		Ввод и вывод данных
Процесс		Вычислительные действия
Решение		Наличие условия в алгоритме

*Окончание таблицы*

Наименование	Символ	Выполняемые действия
Граница цикла		Наличие цикла в алгоритме
Предопределенный процесс		Наличие подпрограммы
Соединитель		Разрыв алгоритма (соединительные блоки)
Комментарий		Внесение комментариев

**Терминатор (пуск-остановка).** Элемент отображает вход из внешней среды или выход из нее (наиболее частое применение – начало и конец алгоритма). Внутри фигуры записывается соответствующее действие – начало/конец.

**Данные (ввод-вывод).** Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод). Данный символ не определяет типа носителя данных.

**Процесс.** Выполнение одной или нескольких операций, обработка данных любого вида (изменение значения данных, формы представления, расположения). Внутри фигуры записывают непосредственно сами операции. В этом блоке знак « $\Rightarrow$ » означает не математическое равенство, а операцию присваивания.

**Решение.** Отображает решение или функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента. Вход в элемент обозначается линией, входящей обычно в верхнюю вершину элемента. Если выходов два или три, то обычно каждый выход обозначается линией, выходящей из оставшихся вершин (боковых и нижней). Если выходов больше трех, то их следует показывать одной линией,

выходящей из вершины (чаще нижней) элемента, которая затем разветвляется. Соответствующие результаты вычислений могут записываться рядом с линиями, отображающими эти пути.

**Граница цикла.** Условия цикла и приращения записываются внутри символа цикла – в зависимости от типа организации цикла. Для изображения итерационного цикла на блок-схеме вместо данного символа используют символ решения, указывая в нем условие, а одну из линий выхода замыкают выше в блок-схеме (перед операциями цикла).

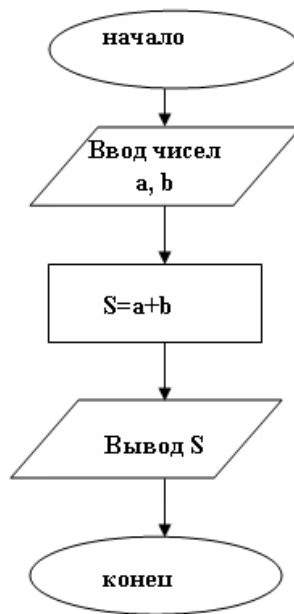
**Предопределенный процесс.** Символ отображает выполнение процесса, состоящего из одной или нескольких операций, который определен в другом месте программы (в подпрограмме, модуле). Внутри символа записывается название процесса и передаваемые в него данные.

**Соединитель.** Используется для обрыва линии и продолжения ее в другом месте (пример: разделение блок-схемы, не помещающейся на листе). Соответствующие соединительные символы должны иметь одно (при том уникальное) обозначение.

**Комментарий.** Позволяет включать в схемы алгоритмов пояснения к функциональным блокам. Частое использование комментариев нежелательно, т. к. это усложняет (загромождает) схему, делает ее менее наглядной.

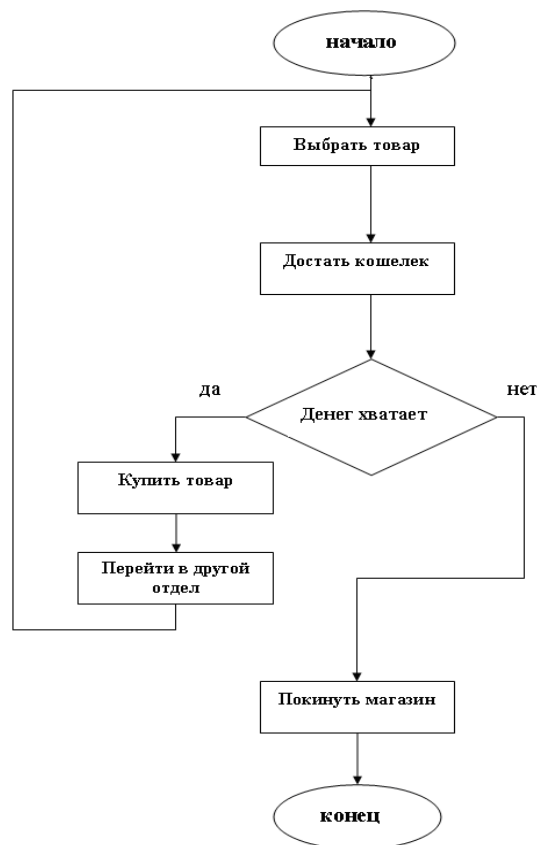
Пр и м е р 7

Алгоритм сложения двух чисел.



### Пример 8

Алгоритм совершения покупок в магазине.



Составление блок-схемы алгоритма является важным и необходимым этапом решения задачи. Но, т. к. в таком виде алгоритм не может быть выполнен непосредственно ЭВМ, этот этап является промежуточным, значительно облегчающим процесс составления программы.

## Программа

Программа – это алгоритм, записанный в виде последовательности команд, понятных ЭВМ (машинных команд). При записи алгоритмов в виде программ для ЭВМ используются языки программирования – системы кодирования предписаний и правила их использования. Такие языки являются искусственными языками со строго определенными синтаксисом и пунктуацией. Они не допускают свободного толкования для своих конструкций, как это характерно для естественного языка. Существует большое количество языков программирования, предназначенных для решения прикладных задач.

Для записи алгоритмов в виде программ характерна высокая степень формализации. Перед составлением программ чаще всего используются словесно-формульный или графический способы.

### Пример 9

Алгоритм сложения двух чисел. Программа, записанная на языке Qbasic:

```
Rem Вычисление суммы двух чисел  
Input «Введите число a»; a  
Input «Введите число b»; b  
Print «Сумма a+b=»; a+b  
End
```

## ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ (ВИДЫ АЛГОРИТМОВ)

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных базовых (т. е. основных) элементов. Эти элементарные шаги объединяются в алгоритмические конструкции.

В зависимости от особенностей своего построения алгоритмы можно разделить на следующие группы:

- 1) линейные (последовательные);

- 2) разветвляющиеся;
- 3) циклические;
- 4) рекурсивные.

Разнообразие алгоритмов определяется тем, что любой алгоритм состоит из фрагментов, каждый из которых представляет собой алгоритм одного из указанных видов. Поэтому важно знать структуру каждого из алгоритмов и принципы их составления.

Для решения любой задачи могут быть построены несколько алгоритмов, приводящих к получению результата ее решения. Из всех возможных алгоритмов следует выбирать наилучший по разным критериям: по точности решения задачи, временным затратам, количеству этапов в алгоритме, их простоте и т. д.

## **Линейная алгоритмическая конструкция**

*Линейным* называется алгоритм, в котором все этапы решения задачи выполняются ровно один раз и строго последовательно. Т. е. линейный (последовательный) алгоритм выполняется в естественном порядке его написания и не содержит разветвлений и повторений.

Примерами линейных алгоритмов являются: алгоритм отпирания дверей – достать ключ, вставить ключ в замочную скважину, открыть замок; алгоритм заваривания чая – достать чайник, насыпать в него чай, залить кипятком, настоять 5-10 мин.

Линейный алгоритм применяется при вычислении арифметического выражения, если в нем используются только простейшие алгебраические действия. Структура такого алгоритма представлена на рис.1.

**Блок-схема**

**Псевдокод**

[действие 1]

[действие 2]

.....



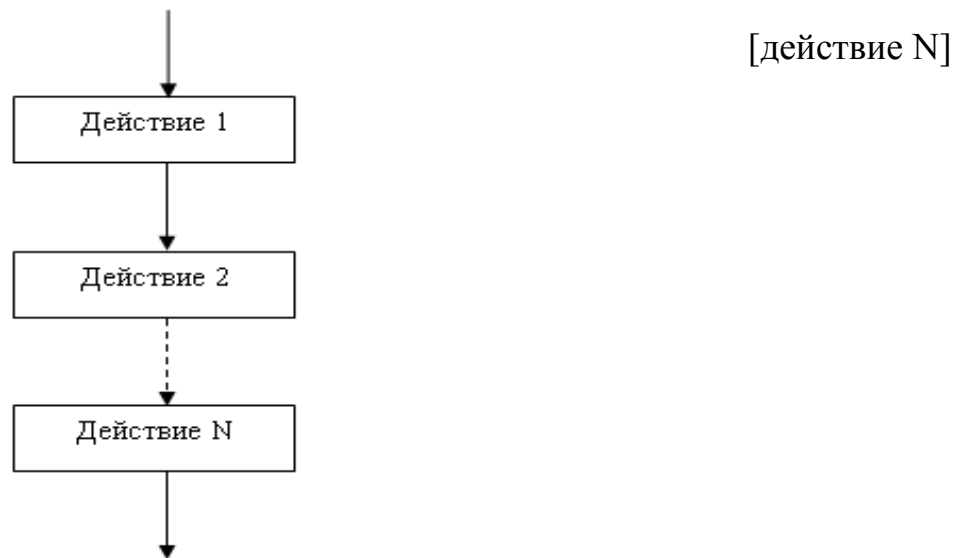


Рис. 1. Линейный алгоритм

Прямоугольник, изображенный на рисунке, может представлять как одну команду, так и множество операторов, необходимых для выполнения обработки данных.

Итак, при разработке линейного алгоритма необходимо учитывать:

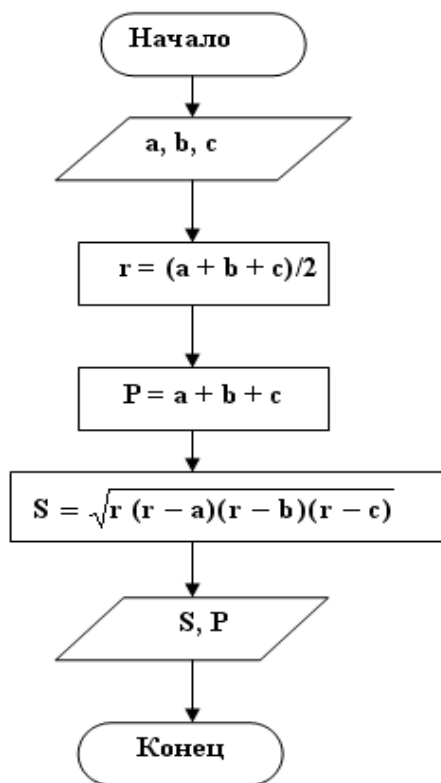
- что данный вид алгоритма является простейшим;
- он чаще используется для реализации простых вычислений по формулам;
- инструкции в нем выполняются последовательно, одна за другой.

### З а д а ч а 1

Составить алгоритм вычисления площади и периметра треугольника, если известны длины трех его сторон.

Входные данные:  $a$ ,  $b$ ,  $c$  (длины сторон треугольника);

Выходные данные:  $S$ ,  $P$  (площадь и периметр треугольника).



### Псевдокод

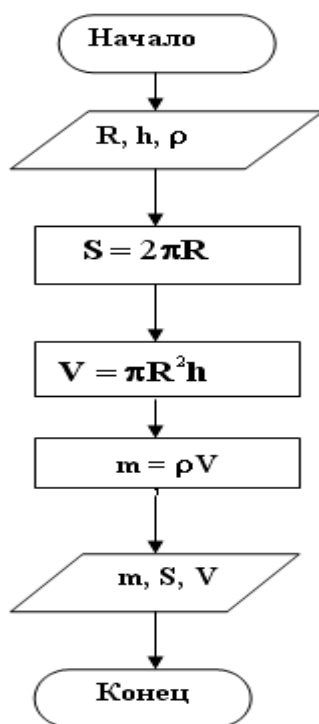
1. Ввод чисел  $a, b, c$
2. Вычисление полупериметра
 
$$r = \frac{a + b + c}{2}$$
3. Вычисление периметра
 
$$P = a + b + c$$
4. Вычисление площади
 
$$S = \sqrt{r(r - a)(r - b)(r - c)}$$
5. Вывод  $S, P$
6. Конец

### Задача 2

Составить алгоритм вычисления объема, массы и площади основания цилиндрического тела, если известны его плотность и геометрические размеры: радиус основания и высота.

Входные данные:  $R$  (радиус основания цилиндра),  $h$  (высота цилиндра),  $\rho$  (плотность материала);

Выходные данные:  $m$  (масса),  $V$  (объем),  $S$  (площадь основания).



## Псевдокод

1. Ввод чисел  $R, h, \rho$
2. Вычисление площади основания  
 $S = 2\pi R$
3. Вычисление объема  
 $V = \pi R^2 h$
4. Вычисление массы  
 $m = \rho V$
5. Вывод  $m, S, V$
6. Конец

## Разветвляющаяся алгоритмическая конструкция

На практике редко удастся представить схему алгоритма решения задачи в виде линейной структуры. Задача может содержать условие, в зависимости от которого вычислительный процесс идет по той или иной ветви, т. е. появляется **ветвление** алгоритма.

**Разветвляющейся** (ветвящейся) называется конструкция, которая обеспечивает выбор между двумя альтернативами в зависимости от значения входных данных.

Или **разветвляющийся алгоритм** — это алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

Примеры разветвляющихся алгоритмов: если пошел дождь, то надо открыть зонт; если билет в театр стоит не больше ста рублей, то купить билет и занять свое место в зале, иначе (если стоимость билета больше 100 руб.) вернуться домой. В общем случае количество ветвей в таком алгоритме не обязательно равно двум. При каждом конкретном наборе входных данных разветвляющийся алгоритм сводится к линейному.

Структура ВЕТВЛЕНИЕ существует в двух основных вариантах:

## ВЕТВЛЕНИЕ

Полное

Неполное

### Полное ветвление

*Предполагает выполнение действий для обеих веток в алгоритме:*

**Если** [условие], **то** [действие 1], **иначе** [действие 2]

Структура такого алгоритма представлена на рис.2.

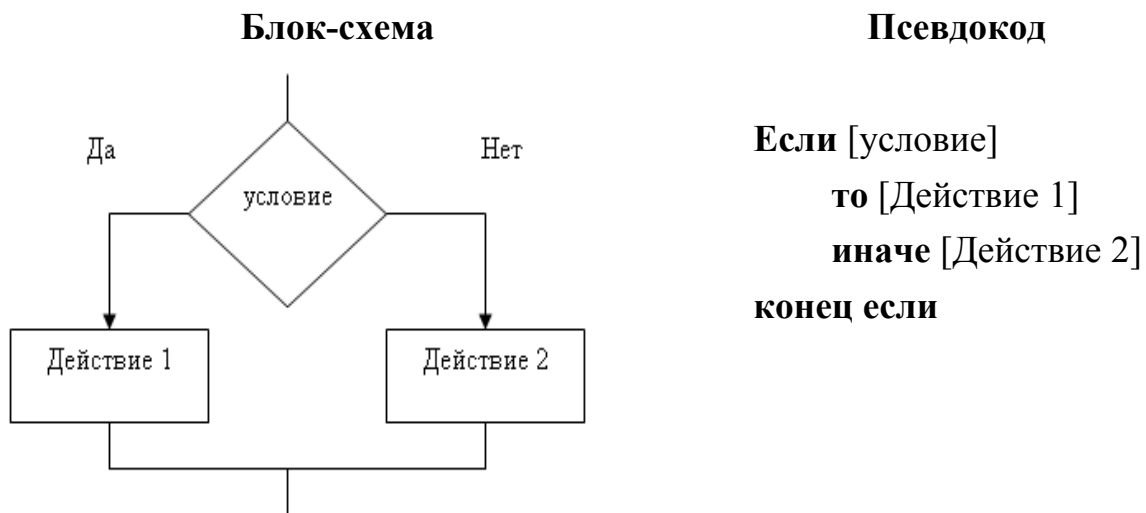


Рис. 2. Полное ветвление

### Неполное ветвление

*Предполагает выполнение действий только на одной ветви алгоритма (вторая отсутствует):*

**Если** [условие], **то** [действие]

Структура такого алгоритма представлена на рис.3.

## Блок-схема

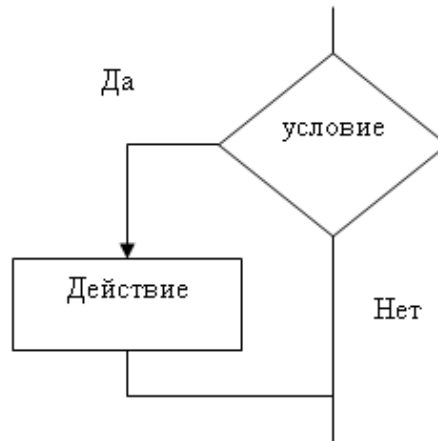


Рис. 3. Неполное ветвление

## Псевдокод

**Если** [условие]  
    **то** [Действие]  
**конец если**

Итак, при разработке разветвляющегося алгоритма необходимо учитывать:

- что данный вид алгоритма применяется при наличии операций условного перехода;
- он чаще используется для вычислений функций, заданных несколькими арифметическими выражениями (формулами);
- инструкции в нем выполняются в зависимости от значения условия.

## З а д а ч а 3

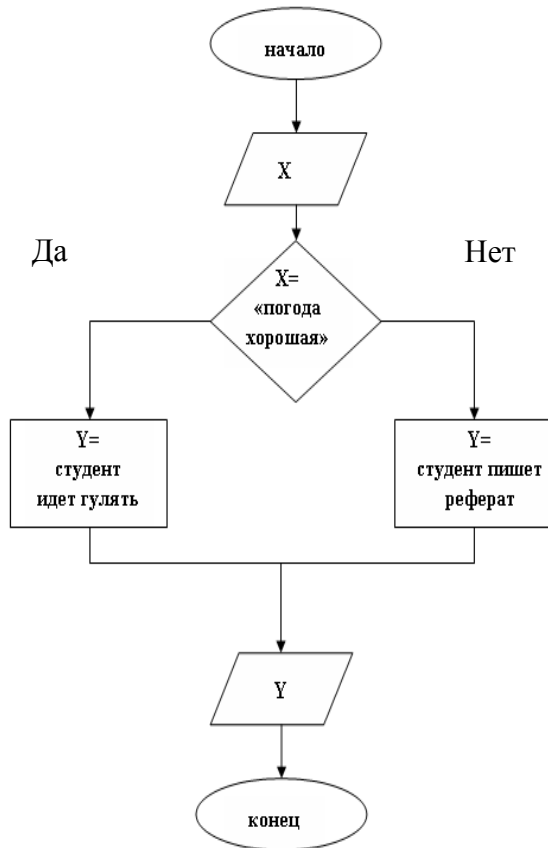
Составить алгоритм планирования выходного дня студентом: если будет хорошая погода, он пойдет гулять, а если плохая – будет писать реферат.

Входные данные:  $x$  (информация о погоде);

Выходные данные:  $y$  (результат прошедшего выходного дня).

## Псевдокод

1. Ввод X (информация о погоде в выходные дни)
2. Проверка условия:  
**Если** X=«Погода хорошая»  
**то** Y=«студент идет гулять»,  
**иначе** Y=«студент пишет реферат»  
**конец если**
3. Вывод Y
4. Конец



### Задача 4

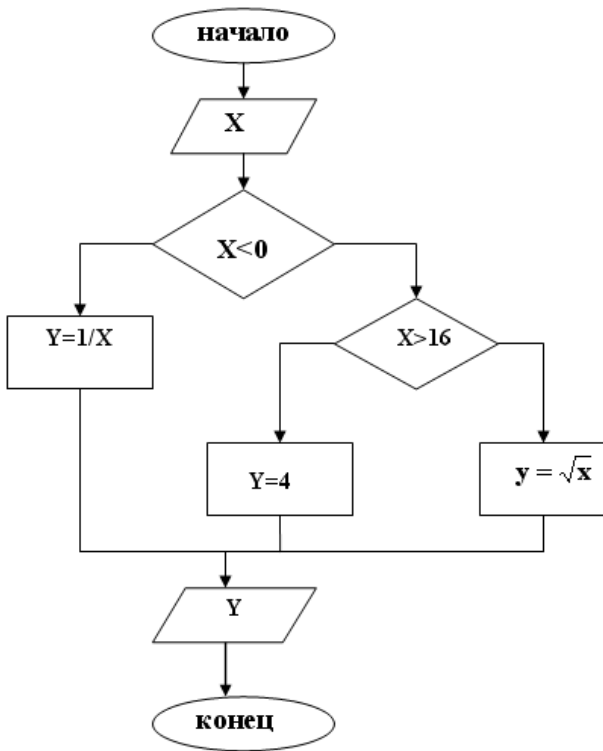
Составить алгоритм вычисления значения функции.

$$y = \begin{cases} \frac{1}{x}, & \text{при } x < 0 \\ \sqrt{x}, & 0 \leq x \leq 16 \\ 4, & \text{при } x > 16 \end{cases}$$

Нахождение значения функции усложняется тем, что на области определения функция состоит из элементарных, непрерывных на заданном участке, функций. Прежде чем вычислить значение  $y$ , необходимо проверить принадлежность введенного  $x$  одному из заданных участков.

Входные данные:  $x$ ;

Выходные данные:  $y$ .



### Псевдокод

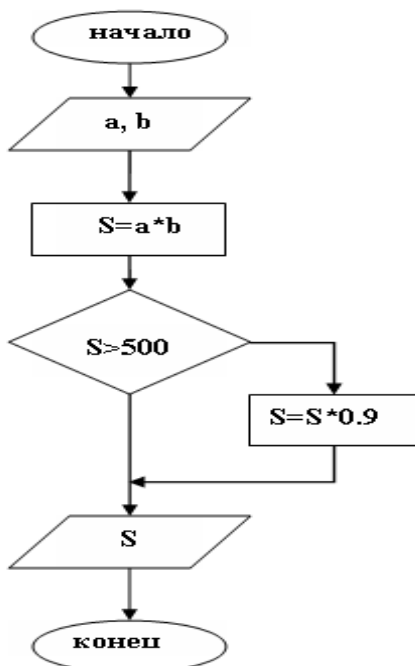
1. Ввод X
2. Проверка условия:  
**Если**  $X < 0$   
**то**  $Y = \frac{1}{x}$   
**иначе**  
**Если**  $X > 16$   
**то**  $Y = 4,$   
**иначе**  $y = \sqrt{x}$   
**конец если**
3. Вывод Y
4. Конец

### Задача 5

Составить алгоритм вычисления стоимости покупки с учетом скидки: при покупке товара на сумму больше 500 руб. предоставляется скидка 10 %.

Входные данные: a (цена единицы товара), b (количество единиц товара);

Выходные данные: s (сумма покупки).



### Псевдокод

1. Ввод чисел a, b
2. Вычисление стоимости покупки  
 $S = a * b$
3. Проверка условия:  
**Если**  $S > 500$   
**то**  $S = S * 0.9$   
**конец если**
4. Вывод S
5. Конец

## Циклическая алгоритмическая конструкция

Часто при решении задач приходится проверять одно условие для нескольких значений или повторять некоторые действия несколько раз. Алгоритмы, отдельные действия которых многократно повторяются, называются *алгоритмами циклической структуры*. Она позволяет существенно сократить объем алгоритма, представить его компактно за счет организации повторений большого числа одинаковых вычислений над разными данными для получения необходимого результата.

**Циклической** называется конструкция, в которой некая, идущая подряд группа действий (шагов), может выполняться несколько раз, в зависимости от входных данных или условия задачи.

Совокупность повторяющихся действий алгоритма называют **циклом**, т. е. **циклические алгоритмы** включают в себя циклы.

Группа повторяющихся действий на каждом шагу цикла называется **телом цикла**.

Примеры циклических алгоритмов: покраска забора, прием документов у абитуриентов в приемной комиссии.

Циклический алгоритм включает в себя:

1. Подготовку цикла – действия, связанные с заданием исходных данных, используемых в цикле;
2. Тело цикла – повторяющиеся действия для вычисления искомых величин, а также подготовка значений, необходимых для повторного выполнения действий в теле цикла;
3. Условия продолжения цикла – действия, определяющие необходимость дальнейшего выполнения тела цикла.

Существует несколько видов циклических конструкций, с помощью которых можно организовать циклы. Их можно классифицировать следующим образом (рис. 4.).



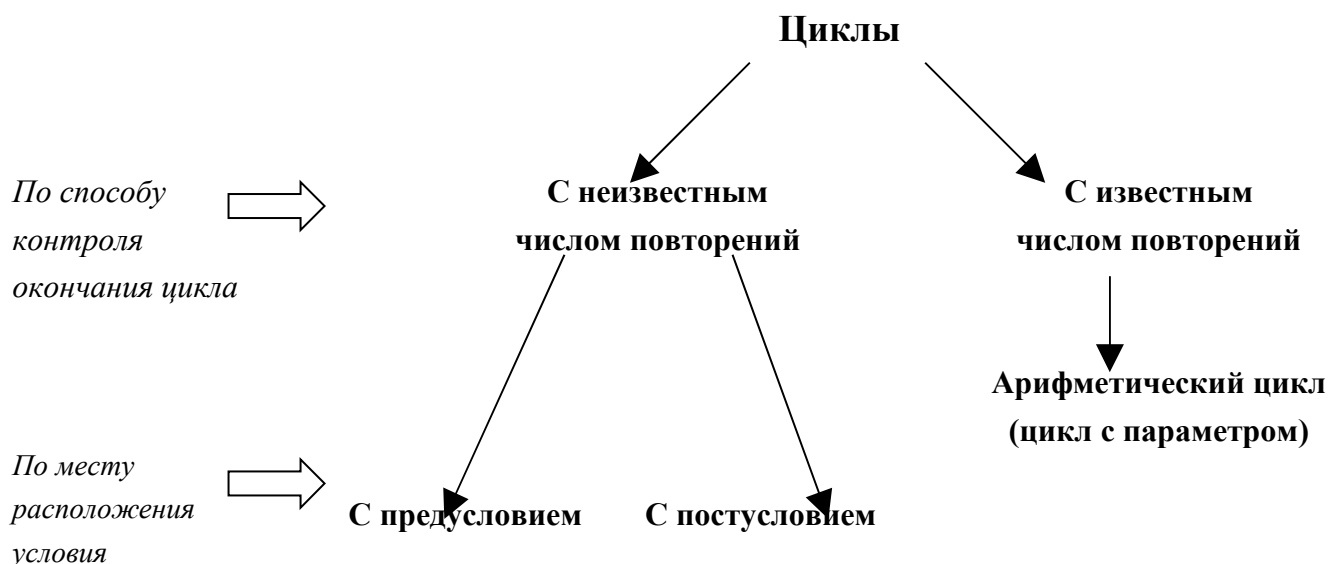


Рис. 4. Классификация циклических конструкций

Циклы, для которых число повторений определяется в ходе выполнения цикла, называются **итерационными** (циклы с неизвестным числом повторений). В этом случае одно повторение цикла называется **итерацией**. Для лучшего понимания их основных отличий друг от друга рассмотрим каждый тип отдельно.

### Арифметический цикл (цикл с параметром)

Предполагает, что число итераций заранее известно. Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.

Цикл типа *Для*:

*Для всех* [параметр цикла] *повторять* [действие]

[параметр цикла] – счетчик количества повторений;

[действие] — тело цикла (последовательность команд, действий).

Количество повторений однозначно определяется правилом изменения параметра, которое задается с помощью начального и конечного значений параметра и шагом его изменения. На первом шаге цикла значение параметра равно  $N$ , на втором –  $N+h$ , на третьем –  $N+2h$  и т.д. На последнем шаге цикла значение параметра не больше  $M$ , но такое, что дальнейшее его изменение приведет к значению большему, чем  $M$ .

$i$  – параметр цикла (является счетчиком количества повторений);

$N$  – начальное значение параметра цикла;

$M$  – конечное значение параметра цикла;

$h$  – шаг, с которым изменяется параметр цикла.

Структура цикла с параметром представлена на рис. 5.

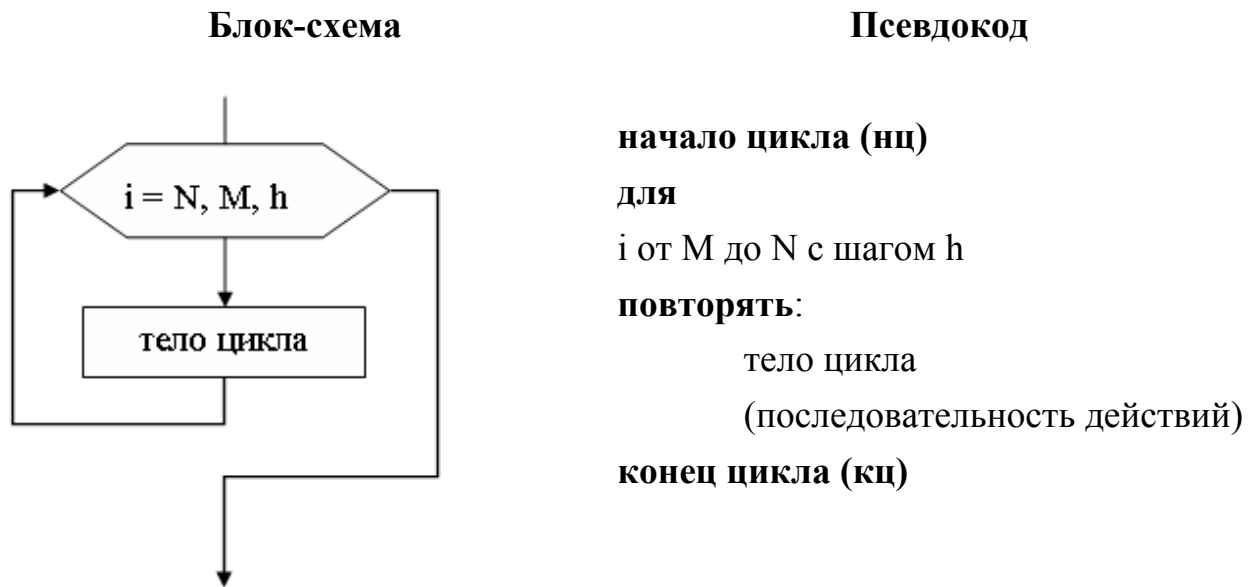


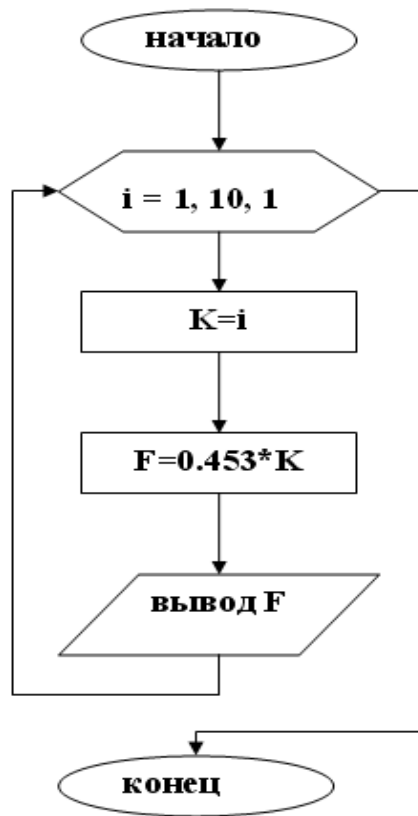
Рис. 5. Цикл с параметром

### Задача 6

Составить алгоритм вывода таблицы соответствия между весом в фунтах и весом в килограммах для значений от 1 до 10 фунтов (1 фунт = 0,453 кг).

Входные данные:  $K$  (вес в килограммах);  $i$  (параметр цикла);

Выходные данные:  $F$  (вес в фунтах).



### Псевдокод

#### 1. Начало цикла

Для  $i = 1, 10, 1$  повторить:

а)  $K = i$

б)  $F = 0,453 * K$

в) Вывод  $F$

**Конец цикла**

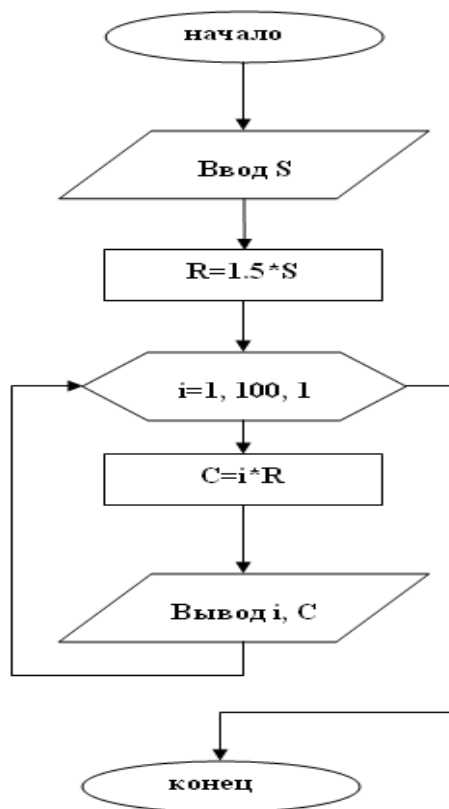
#### 2. Конец

### Задача 7

Составить алгоритм вывода таблицы стоимости поездки на такси в рублях, в зависимости от расстояния (1 км стоит 1,5\$, доллар равен курсу по ЦБ).

Входные данные:  $S$  (курс доллара);  $i$  (количество километров, параметр цикла);

Выходные данные:  $C$  (стоимость поездки).



## Псевдокод

1. Ввод S
2. Вычислить  $R=1.5*S$
3. **Начало цикла**  
 Для  $i = 1, 100, 1$  повторить:
  - а) Вычислить  $C = i * R$
  - б) Вывод  $i, C$**Конец цикла**
4. Конец

## Цикл с предусловием

Предполагает, что число итераций заранее не определено и зависит от входных данных задачи. В данной циклической структуре сначала проверяется значение условия перед выполнением очередного шага цикла.

Цикл типа *Пока*:

*Пока* <условие> выполнять [действие]

<условие> — некоторое проверяемое логическое условие;

[действие] — тело цикла (последовательность команд, действий).

Условие записывается в виде логического выражения, в нем определяется необходимость дальнейшего выполнения повторяющихся действий.

Оператор цикла с предусловием выполняется следующим образом:

- сначала проверяется условие продолжения цикла;
- если это условие истинно, то выполняется тело цикла;
- затем снова проверяется условие продолжения цикла и т. д.;
- если условие продолжения цикла ложно, то происходит выход из цикла.

Особенностью цикла с предусловием является то, что если изначально условие ложно, то тело цикла не выполнится ни разу.

Структура цикла с предусловием представлена на рис. 6.

### Блок-схема

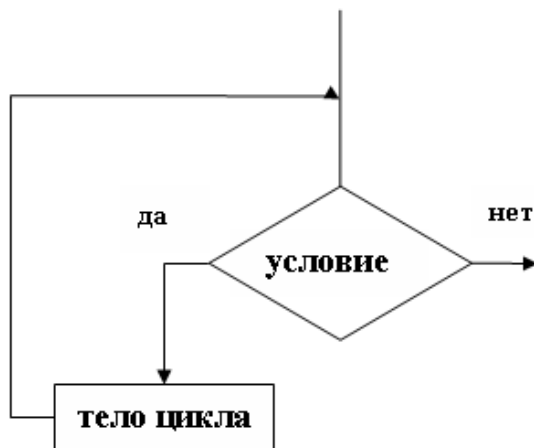


Рис. 6. Цикл с предусловием

### Псевдокод

**начало цикла (нц)**

**пока**

<условие> истинно

**выполнять:**

тело цикла

(последовательность действий)

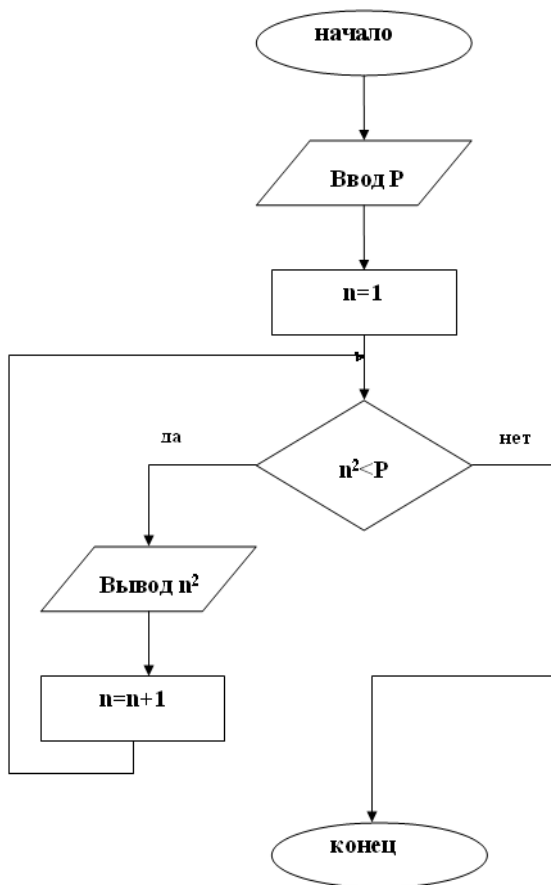
**конец цикла (кц)**

### Задача 8

Составить алгоритм выбора чисел, меньше заданного числа  $P$ , в последовательности квадратов натуральных чисел (1, 4, 9, 25 и т. д.)

Входные данные:  $n$  (натуральное число);  $P$  (заданное число);

Выходные данные:  $n^2$ .



## Псевдокод

1. Ввод  $P$
2.  $n=1$  (значение первого натурального числа)

### 3. Начало цикла

Проверка условия  $n^2 < P$

[Пока оно истинно]

### Выполнять

- а) Вывод  $n^2$
- б)  $n = n + 1$  (переход к следующему натуральному числу)

### Конец цикла

4. Конец

## Цикл с постусловием

Так же, как и в цикле с предусловием, число итераций заранее не определено и зависит от входных данных задачи. Но, в отличие от него, значение условия проверяется после выполнения очередного шага цикла.

### Цикл типа *До*:

*Выполнять* [действие] *до* тех пор пока <условие> ложно

<условие> — некоторое проверяемое логическое условие;

[действие] — тело цикла (последовательность команд, действий)

Условие записывается в виде логического выражения, в нем определяется необходимость дальнейшего выполнения повторяющихся действий.

Оператор цикла с постусловием выполняется следующим образом:

- сначала выполняется тело цикла;
- затем проверяется условие продолжения цикла;
- если это условие истинно, то снова выполняется тело цикла, затем снова проверяется условие и т. д.;

– если условие продолжения цикла ложно, то происходит выход из цикла.

Особенностью цикла с постусловием является то, что цикл выполнится хотя бы один раз до проверки условия.

Структура цикла с постусловием представлена на рис. 7.

### Блок-схема

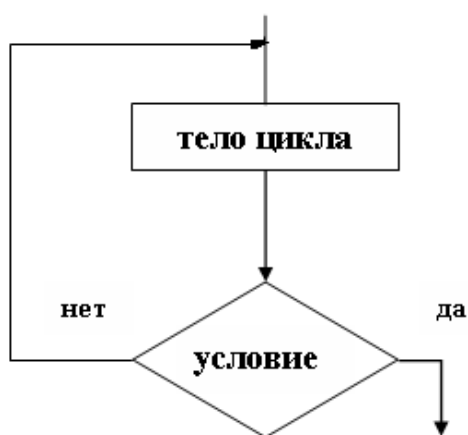


Рис. 7. Цикл с постусловием

### Псевдокод

**начало цикла (нц)**

**выполнять**

тело цикла

(последовательность действий)

до тех пор пока истинно <условие>

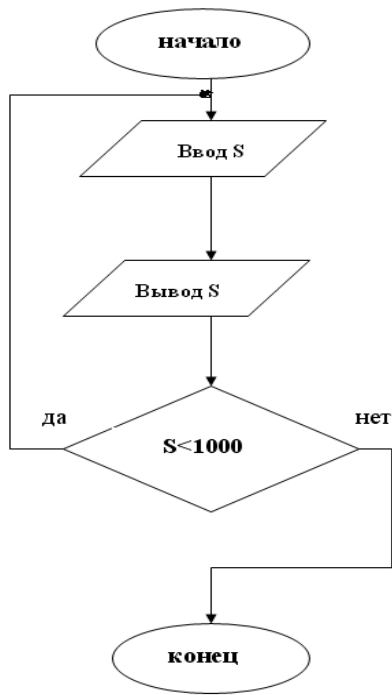
**конец цикла (кц)**

### Задача 9

Составить алгоритм вывода стоимости товаров в чеке до первой суммы, превышающей 1000 руб.

Входные данные: S (стоимость);

Выходные данные: S.



## Псевдокод

### 1. Начало цикла

#### Выполнять:

а) Ввод S

б) Вывод S

#### До

Проверка условия:

$S < 1000$

#### Конец цикла

### 2. Конец

Итак, при разработке циклического алгоритма необходимо учитывать:

– что использование циклов позволяет существенно сократить схему алгоритма;

– при организации цикла следует особое внимание уделить правильному оформлению изменения параметра цикла, потому что ошибка на этом этапе может привести к «заикливлению» вычислений;

– количество повторений в цикле зависит от входных данных или условий задачи;

– для завершения в теле цикла должны быть инструкции, выполнение которых влияет на завершение цикла;

– он чаще используется для табулирования функций, для решения задач с повторными вычислениями.

## Рекурсивная алгоритмическая конструкция

**Рекурсивным** называется алгоритм, который в процессе выполнения на каком-либо шаге прямо или косвенно обращается сам к себе. Как правило, в основе такого алгоритма лежит рекурсивное определение какого-то понятия.

Пример рекурсивного определения – определение факториала числа  $n$ :

$$n! = 1 * 2 * 3 * 4 * \dots * n$$



или

$$\begin{cases} n \neq 1 \wedge p \wedge n \leq 1 \\ n \neq n * (n - 1), \wedge p \wedge n > 1 \end{cases}$$

Т. е. рекурсивный объект частично определяется через себя.

Преимущество рекурсивного определения объекта заключается в том, что такое конечное определение теоретически способно описывать бесконечно большое число объектов.

В рекурсивном определении должно присутствовать ограничение, *граничное условие*, при выходе на которое рекурсивный возврат заканчивается. В случае с вычислением  $n!$  граничным условием является условие  $n \leq 1$ .

Отличие рекурсивного алгоритма от циклического заключается в том, что в определенной точке алгоритма реализуется тот же самый алгоритм.

Хорошо иллюстрирует конструкцию такого типа детская игра-надоелка «Купи слона»:

- Купи слона!
- Все так говорят, а ты купи слона
- и т. д.*
- Купи слона!...
- Спрашивают до тех пор, пока собеседник не ответит:*
- А ты заверни в бумажку! (*граничное условие*).

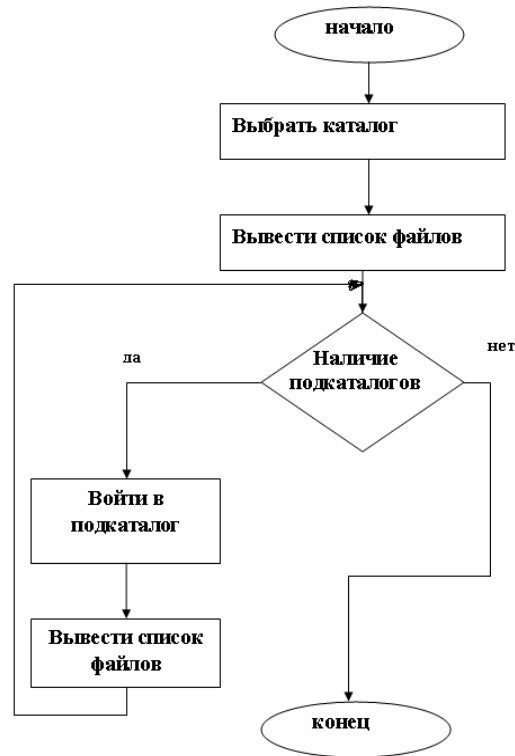
### Задача 10

Составить алгоритм поиска всех файлов с именем «к\*т» в каталоге и во всех подкаталогах этого каталога.

#### Словесный алгоритм

1. Открыть каталог
2. Вывести список всех файлов, удовлетворяющих критерию запроса:  
имя = <к\*т>
3. Если в каталоге есть подкаталоги, то выполнить п. 1, иначе конец.

## Блок-схема



Для того чтобы проверить подкаталоги, алгоритм проверки обращается сам к себе.

## ТИПОВЫЕ ПРИЕМЫ АЛГОРИТМИЗАЦИИ

При решении большинства задач используется определенный набор типовых приемов алгоритмизации. Среди них рассмотрим те, которые наиболее часто применяются при решении практических задач.

**З а м е ч а н и е:** При решении задач, связанных с вычислениями и преобразованиями математических выражений, их следует проанализировать: для всех ли значений переменных их можно вычислить. В алгоритме необходимо предусмотреть предварительную проверку переменных на значения, для которых выражение не может быть определено (подкоренное выражение корня четной степени не должно принимать отрицательные значения, в дроби знаменатель не может быть равен 0, в логарифме основание должно быть положительным и не равным 1 и т. д.)

В блок-схеме такие проверки реализуются с помощью условного блока. Несоблюдение этого правила может привести к критическим ошибкам.

## Комбинированные алгоритмы

На практике часто встречаются задачи, алгоритм решения которых распадается на части, фрагменты и каждый фрагмент представляет собой алгоритм одного из трех описанных типов. Алгоритм, который содержит несколько структур одновременно, называется **комбинированным**.

В теории алгоритмов доказано, что любой, сколь угодно сложный алгоритм может быть составлен из трех основных алгоритмических структур: линейной, ветвления и цикла. Алгоритмы решения сложных задач могут включать все перечисленные структуры либо комбинации некоторых из них.

### Задача 11

Составить алгоритм вычисления  $Z = \min(A - B, \max(C^3, A + B + C))$ .  $A, B, C$  – действительные числа.

Возможны три варианта ответа: или  $(A - B)$ , или  $C^3$ , или  $(A + B + C)$ .

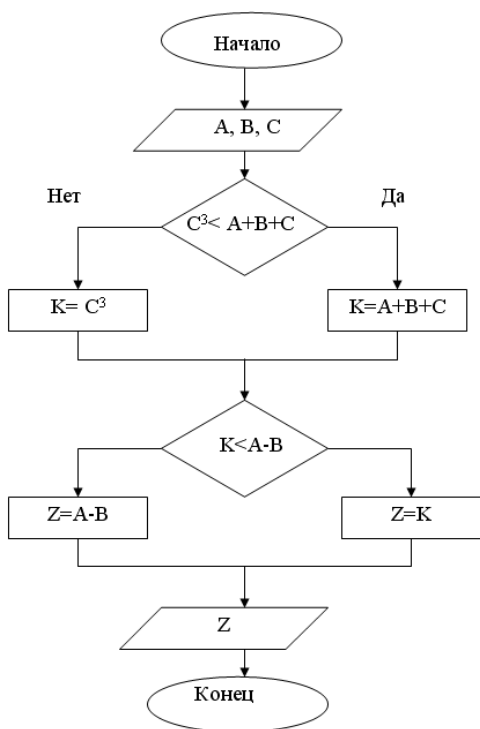
Обозначим:

$$K = \max(C^3, A + B + C) = \begin{cases} C^3, & \text{если } C^3 \geq A + B + C \\ A + B + C, & \text{если } C^3 < A + B + C \end{cases}$$

$$Z = \min(A - B, K) = \begin{cases} A - B, & \text{если } A - B \leq K \\ K, & \text{если } A - B > K \end{cases}$$

Входные данные:  $A, B, C$ ;

Выходные данные:  $K, Z$ .



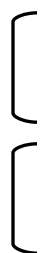
### Псевдокод

1. Ввод A, B
2. Проверка условия:  
**Если**  $C^3 < A + B + C$   
**то**  $K = C^3$   
**иначе**  $K = A + B + C$   
**конец если**
3. Проверка условия:  
**Если**  $K < A - B$   
**то**  $Z = A - B$   
**иначе**  $Z = K$   
**конец если**
4. Вывод Z
5. Конец

## Вложенные циклы

Частным случаем комбинированных алгоритмов являются алгоритмы, в которых необходимо использовать несколько циклов, изменяющихся одновременно. Например, внутри одного цикла могут находиться один или несколько других циклов. Такие комбинации называются **вложенными циклами**.

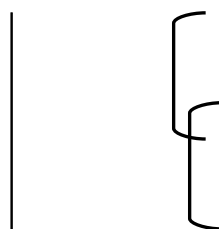
Охватывающий цикл называется **внешним**, а вложенные в него циклы – **внутренними** (вложенными), при этом область действия внутреннего цикла должна полностью находиться в области внешнего цикла (рис. 8, б), т. е. циклы не должны пересекаться (рис. 8, в).



а - последовательные циклы



б - вложенные циклы



в - запрещенные циклы

Рис. 8. Расположение циклов

Правила организации как внешнего, так и внутреннего циклов такие же, как и правила организации простого цикла. Параметры внешнего и внутреннего должны иметь разное имя и изменяться не одновременно, т. е. при одном значении параметра внешнего цикла параметр внутреннего цикла принимает по очереди все свои значения. Перед циклом необходимо задавать начальные значения параметров, а внутри – вычислять текущие.

### Задача 12

Составить алгоритм заполнения таблицы данными о зарплате каждого сотрудника за первый квартал 2010 года.

ФИО сотру дника	Я нварь	Фе враль	М арт
Ивано в	10 00	120 0	1 300
Петро в	11 00	100 0	1 200
Сидор ов	20 00	210 0	2 200
Васеч кин	30 00	310 0	3 300

Обозначим вводимые данные как элементы двумерного массива  $\{A(i;j)\}$ , где  $i = 1, 2, 3, 4$  (номер строки);

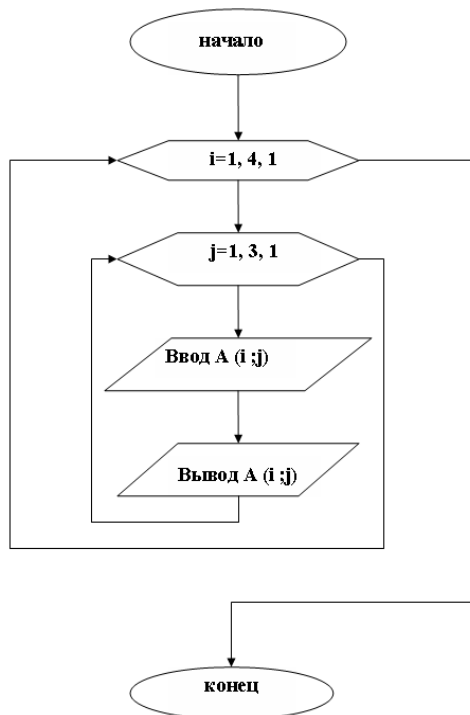
$j = 1, 2, 3$  (номер столбца).

$A(1;1)$	$A(1;2)$	$A(1;3)$
=1000	=1200	=1300
$A(2;1)$	$A(2;2)$	$A(2;3)$
=1100	=1000	=1200
$A(3;1)$	$A(3;2)$	$A(3;3)$
=2000	=2100	=2200
$A(4;1)$	$A(4;2)$	$A(4;3)$
=3000	=3100	=3300

Задача сводится к заполнению двумерного массива.

Входные данные:  $i, j, A(i;j)$ ;

Выходные данные:  $A(i;j)$



### Псевдокод

**Начало цикла**

Для  $i=1, 4, 1$  повторить:

**Начало цикла**

Для  $j=1, 3, 1$  повторить:

а) ввод  $A(i;j)$ ;

б) вывод  $A(i;j)$ ;

**Конец цикла**

**Конец цикла**

**Конец**

## Вычисление суммы и произведения

Для построения алгоритма решения таких задач используется цикл с параметром, т. к. число шагов цикла известно.

### Суммирование

$$S = a_1 + a_2 + a_3 + a_4 + \dots$$

При вычислении суммы используется прием накопления.

$$S_1 = S + a_1$$

$$S_2 = S + a_1 + a_2 = S_1 + a_2$$

$$S_3 = a_1 + a_2 + a_3 = S_2 + a_3$$

и т. д.

$S$  – начальное значение суммы

$S_i$  – значение суммы на каждом шаге вычисления

$a_i$  – очередное слагаемое

Вычисление суммы сводится к ее накоплению в виде значения переменной в цикле, в котором вычисляются соответствующие слагаемые. При этом вновь вычисленное слагаемое прибавляется к сумме предыдущих слагаемых, т. е. в цикле последовательно вычисляются все промежуточные суммы.

После первого выполнения цикла первая промежуточная сумма должна быть равна значению первого слагаемого. Следовательно, начальное значение  $S$  должно быть равно нулю.

**Правило суммирования:**

начальное значение суммы  $S=0$

в теле циклической конструкции выполняется команда

$$S = S + \langle \text{слагаемое} \rangle$$

**Формула для накопления суммы**

$$S = S + a,$$

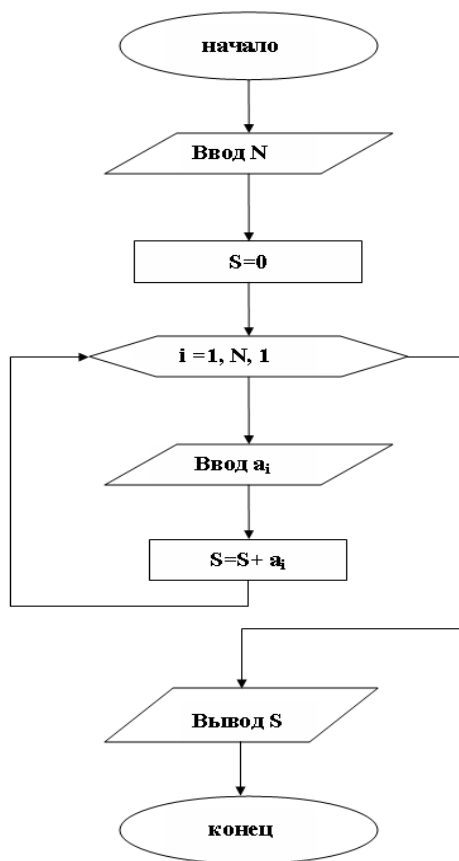
где  $a$  – очередное слагаемое;  $S$  – промежуточная сумма.

**З а д а ч а 13**

Составить алгоритм вычисления общей суммы выплаченной премии всем сотрудникам фирмы.

Входные данные:  $N$  (количество сотрудников фирмы);  $a_i$  (размер премии каждого сотрудника);  $i$  (параметр);

Выходные данные:  $S$  (общая сумма выплат).



### Псевдокод

1. Ввод N
2. Начальное значение суммы  
 $S=0$
3. **Начало цикла**  
Для  $i=1, N, 1$  повторить:
  - а) ввод  $a_i$
  - б) вычисление текущей суммы  $S=S+ a_i$
- Конец цикла**
4. Вывод S
5. Конец

### Произведение

$$P=a_1*a_2*a_3*a_4*.....$$

При вычислении произведения используется тот же прием накопления.

$$P_1=P*a_1$$

$$P_2=P*a_1*a_2=P_1*a_2$$

$$P_3=a_1*a_2*a_3=P_2*a_3$$

и т. д.

$P$  – начальное значение произведения

$P_i$  – значение произведения на каждом шаге вычисления

$a_i$  – очередной множитель

Вычисление произведения сводится к его накоплению в цикле в виде значения переменной, при этом в цикле вычисляются последовательно все промежуточные произведения.

После первого выполнения цикла первая промежуточная величина должна быть равна значению первого множителя. Следовательно, начальное значение  $P$ , которое задается перед циклом, должно быть равно единице.

### Правило умножения:

начальное значение произведения  $P=1$



в теле циклической конструкции выполняется команда

$$P = P * \langle \text{множитель} \rangle$$

### Формула для накопления произведения

$$P = P * a,$$

где  $a$  – очередной сомножитель;  $P$  – промежуточное произведение.

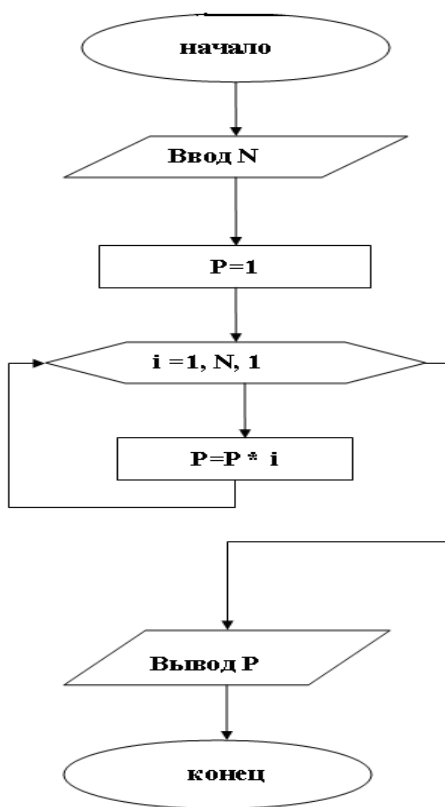
### Задача 14

Составить алгоритм вычисления  $n!$ .

$$(n! = 1 * 2 * 3 * 4 * \dots * n)$$

Входные данные:  $n$  (натуральное число);  $i$  (параметр);

Выходные данные:  $P = n!$



### Псевдокод

1. Ввод  $N$
2. Начальное значение произведения  $P=1$
3. **Начало цикла**  
Для  $i=1, N, 1$  повторить:  
 $P = P * a$   
(Вычисление текущего произведения)
- Конец цикла**
4. Вывод  $P$
5. Конец

### Подсчет количества элементов

При вычислении количества элементов используется прием накопления, как и при вычислении суммы и произведения.

Подсчет количества элементов сводится к вычислению количества натуральных чисел в последовательности 1, 2, 3, 4 и т. д.

**Счетчик** искомым элементов обозначается **К**.

Процесс является циклическим, так как каждый раз совершается одно и то же действие: увеличение предыдущего натурального числа на 1.

При первом подсчете должно получиться значение  $K$ , равное единице. Следовательно, начальное значение счетчика должно быть равно нулю.

**Правило счетчика:**

начальное значение счетчика  $K=0$

– в теле циклической конструкции выполняется команда

$$K = K+1$$

если не задано дополнительных условий

**Формула для накопления счетчика:**

$$K=K+1$$

где  $K$  – промежуточное значение счетчика.

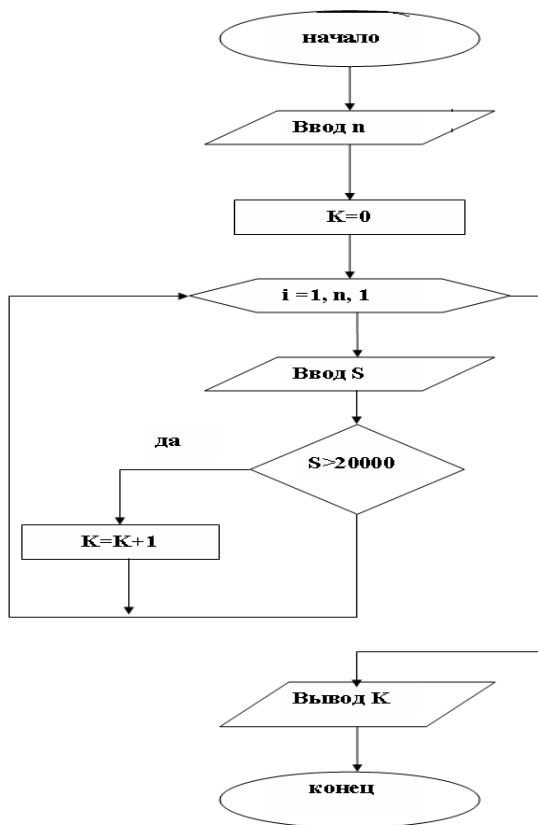
**Задача 15**

Составить алгоритм подсчета количества сотрудников фирмы, зарплата которых превышает 20 тыс. руб.

(Составить алгоритм подсчета количества чисел больших 20 среди  $n$  заданных).

Входные данные:  $n$  (число сотрудников);  $S$  (размер зарплаты);  $i$  (параметр);

Выходные данные: счетчик  $K$  (количество сотрудников).



### Псевдокод

1. Ввод n
2. Начальное значение счетчика K=0
3. **Начало цикла**  
 Для i=1, n, 1 повторить:
  - а) ввод S
  - б) проверка условия:  
**Если S > 20000**  
**то K=K+1**  
**конец если****Конец цикла**
4. Вывод K
5. Конец

## Табулирование функций

**Табулирование** функции – это вычисление значений функции при изменении аргумента от некоторого начального значения до некоторого конечного значения с определенным шагом. При этом вид функции и шаг может меняться.

Табулирование применяется для составления всевозможных таблиц, которыми могут быть как абстрактная таблица значений математической функции, так и конкретная таблица стоимости товара или платежей, совершенных абонентом сотового оператора.

Общая математическая формулировка такой задачи:

Вычислить и напечатать таблицу значений аргумента X и функции  $Y = f(x)$  при изменении аргумента X на отрезке [a; b] с шагом h.

a – начало отрезка;

b – конец отрезка;

h – шаг;

n – количество шагов;

$(n+1)$  – количество точек.

Параметры связаны следующей формулой:

$$n + 1 = \frac{b - a}{h}.$$

Результатом должна быть следующая таблица:

X	Y
$X_1 =$	Y
a	1
$X_2 =$	Y
$X_1 + h$	2
...	...
$X_n =$	Y
b	n

Алгоритм табулирования содержит все основные конструкции: линейную, ветвление, цикл. В общем виде алгоритм можно описать так:

1. Определяется переменная (обычно X);
2. Перед циклом задается начальное значение переменной, условием окончания цикла является достижение переменной конечного значения;
3. В теле цикла на каждом шаге вычисляется значение функции Y, зависящее от переменной X (формируется строка таблицы);
4. В конце каждого шага значение переменной изменяется на h, где h – заданный шаг изменения, т. е.  $X = X + h$ .

### Табулирование функции, заданной одной формулой

$$Y=f(X)$$

Функция, как правило, определена на заданном участке. Требуется n раз произвести вычисления одного выражения при разных значениях X.

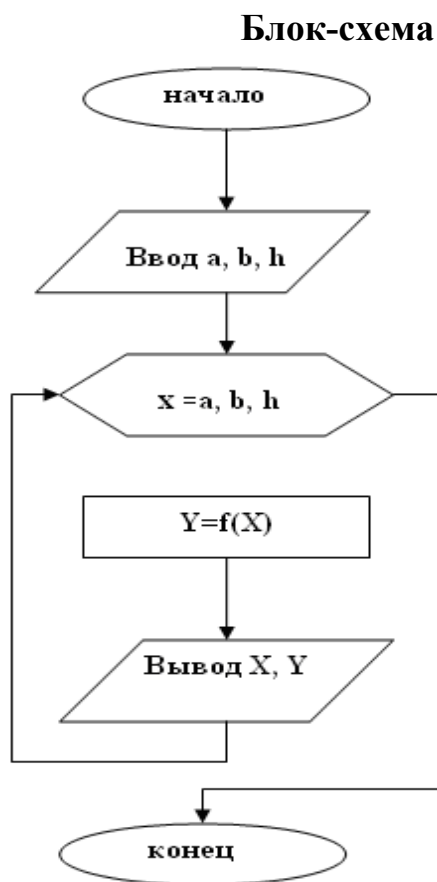
Алгоритм решения такой задачи состоит из следующих основных этапов:

1. Ввод исходных данных a, b, h;

2. Организация цикла по  $X$  от начального значения  $a$  до конечного значения  $b$  с шагом  $h$ ;
3. Вычисление  $Y = f(x)$ ;
4. Печать  $X, Y$ ;
5. Конец цикла.

В зависимости от условий конкретной задачи, к перечисленным этапам могут добавляться другие вычисления.

Блок-схему и псевдокод такой задачи в общем случае можно представить в следующем виде:



**Псевдокод**

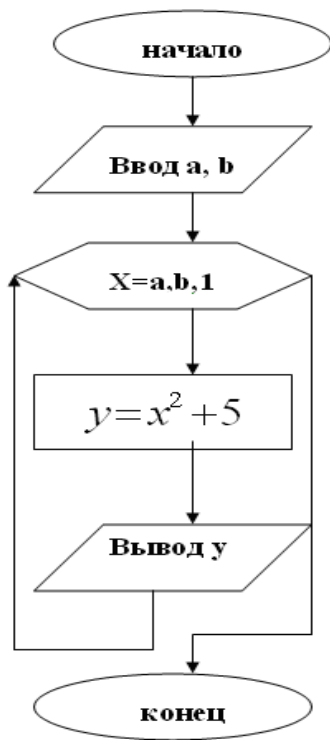
1. Ввод  $a, b, h$
2. **Начало цикла**  
 Для  $x = a, b, h$  повторить:
  - а) вычислить  $y = f(x)$
  - б) Вывод  $x, y$
- Конец цикла**
3. Конец

### Задача 16

Составить алгоритм табулирования функции  $y = x^2 + 5$  на отрезке  $[a; b]$ . Значения  $a, b$  могут изменяться.

Входные данные:  $x$  (переменная – параметр цикла);  $a$  (начальное значение параметра);  $b$  (конечное значение параметра);

Выходные данные:  $y$ .



## Псевдокод

Ввод a, b

**Начало цикла**

Для x=a, b, 1 повторить:

а) вычислить  $y = x^2 + 5$

б) вывод y

**Конец цикла**

Конец

## Табулирование функции, заданной несколькими формулами

$$y = \begin{cases} f_1(x), & x \leq x_1 \\ f_2(x), & x_1 < x \leq x_2 \\ \dots & \\ f_n(x), & x > x_n \end{cases}$$

Функция разделена на промежутки, на которых она задана элементарными функциями. Каждая из этих функций, как правило, определена на заданном интервале.

Такая задача отличается тем, что необходимо, изменяя значение аргумента, проверять, в какой из определенных промежутков он попадает, в качестве значения функции  $Y$  выбирать формулу, соответствующую промежутку.

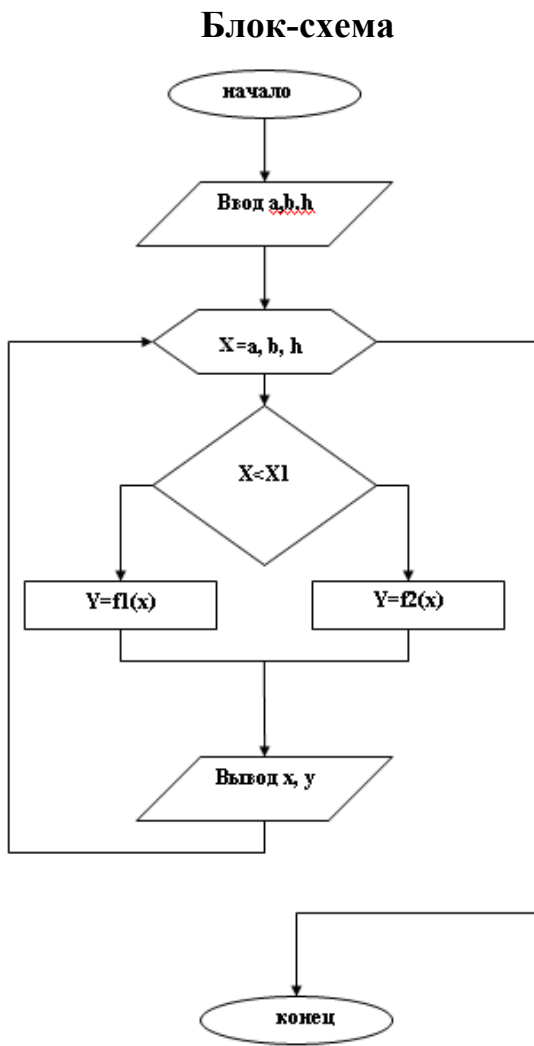
Алгоритм решения такой задачи состоит из следующих основных этапов:

1. Ввод исходных данных a, b, h;
2. Организация цикла по X от начального значения a до конечного значения b с шагом h;
3. Проверка условия: определяется интервал, в котором находится X;

4. Вычисление  $Y_1 = f_1(x)$ ;
5. При необходимости пп. 3–4 повторяются;
6. Печать  $X, Y$ ;
7. Конец цикла.

В зависимости от условий конкретной задачи, к перечисленным этапам могут добавляться другие вычисления.

Блок-схему и псевдокод такой задачи в общем случае можно представить в следующем виде:



**Псевдокод**

1. Ввод  $a, b, h$
2. **Начало цикла**  
 Для  $x=a, b, h$  повторить:
  - а) проверка условия:  
**Если**  $X < X1$   
**то**  $Y=f1(x)$ ,  
**иначе**  $Y=f2(x)$   
**конец если**
  - б) вывод  $X, Y$**Конец цикла**
3. Конец

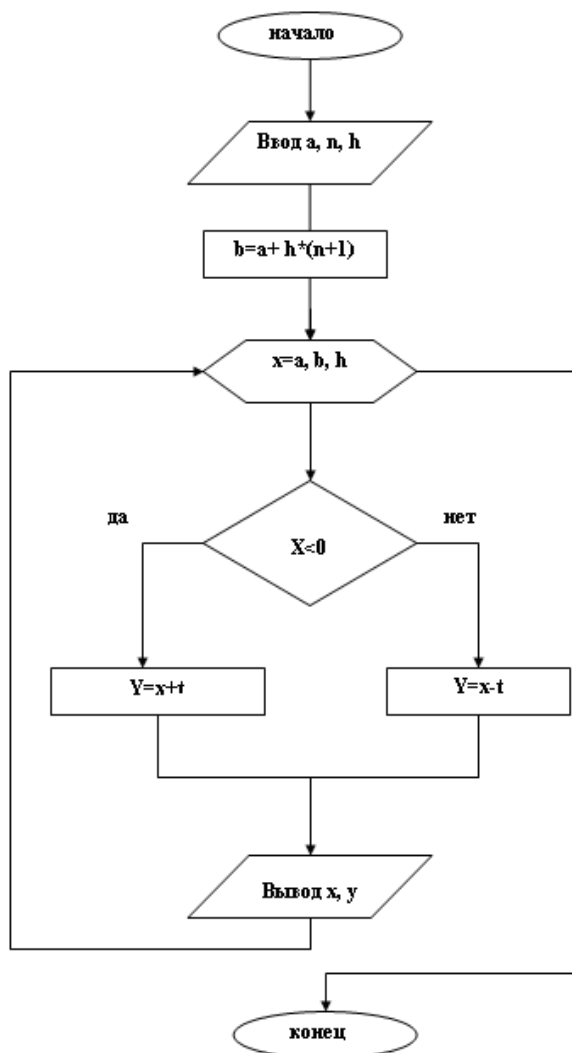
Задача 17

Составить алгоритм табулирования функции  $y = \begin{cases} x + t, & \text{при } x \geq 0 \\ x - t, & \text{при } x < 0 \end{cases}$  на отрезке

$[a; b]$ , если  $a = -2$ ,  $h = 0,2$ ,  $n = 11$ .

Входные данные:  $x$  (переменная – параметр цикла);  $a$  (начальное значение параметра);  $n$ ;  $h$ ;

Выходные данные:  $y$ .



### Псевдокод

1. Ввод  $a, n, h$
2. Вычислить  $b$
3. **Начало цикла**

Для  $x=a, b, h$  повторить:

а) Проверка условия:

**Если**  $X < 0$ ,  
**то**  $Y = x + t$ ,  
**иначе**  $Y = x - t$

**конец если**

б) вывод  $X, Y$

**Конец цикла**

4. **Конец**

Приведенные алгоритмы и способы решения задач являются рациональными, но не претендуют на то, чтобы считаться наилучшими. Как уже отмечалось, любая задача может быть разбита на отдельные этапы, реализация которых осуществляется наиболее удобным способом для каждого случая.



## ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

### Задачи на линейный алгоритм

1. Составить алгоритм вычисления значения функции  $y=7x+5$  при любом значении  $x$ .
2. Составить алгоритм вычисления периметра квадрата, если известна его сторона.
3. Составить алгоритм вычисления длины окружности, если известен ее радиус.
4. Составить алгоритм вычисления площади окружности, если известен ее диаметр.
5. Составить алгоритм вычисления гипотенузы прямоугольного треугольника, если известны его катеты.
6. Составить алгоритм вычисления периметра прямоугольного треугольника, если известны его катеты.
7. Составить алгоритм вычисления периметра прямоугольника и его диагонали, если известны его стороны.
8. Составить алгоритм вычисления площади кольца, если известны радиуса внешней и внутренней окружности.
9. Составить алгоритм вычисления площади поверхности и объема куба, если известно его ребро.
10. Составить алгоритм вычисления периметра равнобедренной трапеции, если известны ее основания и высота.
11. Составить алгоритм определения плотности тела, если известны его объем и масса.
12. Составить алгоритм вычисления суммы, разности, произведения и частного двух чисел.
13. Составить алгоритм вычисления среднего арифметического и среднего геометрического двух положительных чисел.
14. Составить алгоритм вычисления плотности населения в государстве, если известны его площадь и количество жителей.
15. Составить алгоритм вычисления площади поверхности и объема прямоугольного параллелепипеда, если известны его ребра.

## Задачи на разветвляющийся алгоритм

1. Составить алгоритм решения задачи для определения максимального и минимального значения из двух различных вещественных чисел.
2. Составить алгоритм решения задачи: впишется ли круг в квадрат, если известны сторона квадрата и радиус круга.
3. Составить алгоритм решения задачи для вычисления значения функции

$$y = \begin{cases} x^2, & \text{при } x \geq 0 \\ \frac{1}{x^2}, & \text{при } x < 0. \end{cases}$$

4. Составить алгоритм решения задачи для определения впишется ли квадрат в круг, если известны сторона квадрата и радиус круга.
5. Составить алгоритм решения задачи для определения большего из двух вещественных чисел.
6. Составить алгоритм решения задачи для вычисления значения функции

$$y = \begin{cases} x^2, & \text{при } x \geq 0 \\ \frac{1}{x^2}, & \text{при } x < 0. \end{cases}$$

7. Составить алгоритм решения задачи для определения меньшего из двух вещественных чисел.
8. Составить алгоритм решения задачи для определения большего расстояния из двух: одно указано в километрах, а другое в футах (1 фут = 0,45 м).
9. Составить алгоритм решения задачи для вычисления значения функции

$$y = \begin{cases} a^2 + x^2, & \text{при } x \geq 1 \\ a^2 - x^2, & \text{при } x < 1. \end{cases}$$

10. Составить алгоритм вычисления частного двух чисел.

11. Составить алгоритм решения задачи для определения большей скорости: одно значение указано в километрах в час, а другое в метрах в секунду ( $1 \text{ м/с} = 3,6 \text{ км/ч}$ ).
12. Составить алгоритм решения задачи для определения большей площади, если известны радиус круга и сторона квадрата.
13. Составить алгоритм решения задачи для вычисления значения функции

$$y = \begin{cases} \sin x, & \text{при } x \leq \frac{\pi}{4} \\ \cos x, & \text{при } x > \frac{\pi}{4} \end{cases}$$

14. Составить алгоритм решения задачи для определения большего из двух значений: одно указано в миллиметрах, а другое в дюймах ( $1 \text{ дюйм} = 25,4 \text{ мм}$ ).
15. Составить алгоритм решения задачи для определения большей плотности материалов двух тел, если известны их объемы и массы.

### Задачи на цикл с параметром

1. Составить алгоритм вывода таблицы перевода расстояния в дюймах в сантиметры для значений 10, 11, ..., 22 дюйма ( $1 \text{ дюйм} = 25,4 \text{ мм}$ ).
2. Составить алгоритм вывода таблицы перевода перевода 1, 2, ..., 20 долларов США в рубли по текущему курсу (значение курса вводится произвольно).
3. Составить алгоритм вывода значений  $e^1, e^2, \dots, e^{20}$ .
4. Составить алгоритм вывода следующих чисел: 1.1, 2.1, ..., 21.1.
5. Составить алгоритм вывода следующих чисел: 2.1, 2.2, ..., 2.9.
6. Составить алгоритм вывода двадцати первых четных чисел.
7. Составить алгоритм вывода пятнадцати первых нечетных чисел.
8. Составить алгоритм вывода значений следующих чисел:  $\sqrt{2}, \sqrt{4}, \dots, \sqrt{20}$ .
9. Составить алгоритм вывода стоимости 2, 3, ..., 10 кг конфет (цена 1 кг конфет вводится произвольно).
10. Составить алгоритм табулирования функции  $y = \sqrt{x}$  на отрезке  $[a; b]$ . Значения  $a, b$  могут изменяться.

11. Составить алгоритм табулирования функции  $y = e^x - 1$  на отрезке  $[a; b]$ .  
Значения  $a, b$  могут изменяться.
12. Составить алгоритм табулирования функции  $y = \frac{1}{2x}$  на отрезке  $[a; b]$ .  
Значения  $a, b$  могут изменяться.
13. Составить алгоритм табулирования функции  $y = (x - 1)^2$  на отрезке  $[a; b]$ .  
Значения  $a, b$  могут изменяться.
14. Составить алгоритм вычисления значения выражения  $Y$  для значений  $x$ ,  
равных  $1, 2, \dots, 20$ :  $y = 2t^2 + 2t + 2, t = 1 + x$ .
15. Составить алгоритм вычисления значения выражения  $Z$  для значений  $x$ ,  
равных  $2, 4, \dots, 20$ :  $z = 8f^3 - f, f = 2x$ .

### **Задачи на цикл с условием**

1. Составить алгоритм выбора чисел, меньше заданного числа  $P$  в последовательности  $\sqrt{2}, \sqrt{3}, \sqrt{4}$  и т. д.
2. Даны два целых числа  $A$  и  $B$  ( $A < B$ ). Составить алгоритм вывода всех целых чисел, расположенных между данными числами (не включая сами эти числа), в порядке их возрастания.
3. Даны два целых числа  $A$  и  $B$  ( $A < B$ ). Составить алгоритм вывода всех целых чисел, расположенных между данными числами (не включая сами эти числа), в порядке их убывания.
4. Дано целое число  $N$  ( $> 1$ ). Составить алгоритм вывода наименьшего целого  $K$ , при котором выполняется неравенство  $3K > N$  и самого значения  $3K$ .
5. Дано целое число  $N$  ( $> 1$ ). Составить алгоритм вывода наибольшего целого  $K$ , при котором выполняется неравенство  $3K < N$ .
6. Дано натуральное число  $N$ . Составить алгоритм получения всех натуральных чисел, меньше  $N$ .
7. Дано вещественное число  $a$ . Составить алгоритм вывода всех натуральных чисел  $n$ , при которых выполняется условие  $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} > a$ .
8. Дано число  $n$ . Составить алгоритм поиска первого натурального числа, квадрат которого больше  $n$ .

9. Составить алгоритм вывода минимального числа больше 200, которое нацело делится на 17.
10. Составить алгоритм поиска максимального из натуральных чисел, не превышающих 600, которое нацело делится на 28.
11. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10 % от пробега предыдущего дня. Составить алгоритм определения, в какой день он пробежит больше 20 км.
12. Составить алгоритм вычисления суммы ряда  $y = \frac{1}{2} + \frac{3}{4} + \dots + \frac{2n-1}{2n}$  с заданной точностью  $\varepsilon = 0,0001$ .
13. Составить алгоритм вычисления суммы ряда  $S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \dots$  с заданной точностью  $\varepsilon = 0,0001$ .
14. Составить алгоритм вывода всех натуральных чисел, кратных одиннадцати, меньше 100.

### Дополнительные задачи

1. Составить алгоритм получения двухзначного числа путем перестановки его цифр.
2. Составить алгоритм вычисления периметра и площади треугольника, заданного координатами своих вершин.
3. Составить алгоритм нахождения числа десятков и единиц в двухзначном числе.
4. Составить алгоритм определения количества метров в указанном числе сантиметров, дециметров, миллиметров.
5. Составить алгоритм решения задачи для вычисления значения функции

$$y = \begin{cases} k \text{ и } p \text{ и } k < x \\ k + x, \text{ и } p \text{ и } k > x \\ \frac{k}{x}, \text{ и } p \text{ и } k = x. \end{cases}$$

6. Составить алгоритм для нахождения произведения двух наименьших чисел из трех заданных.
7. Составить алгоритм нахождения корней квадратного уравнения  $ax^2 + bx + c = 0$ .
8. Составить алгоритм решения задачи для вычисления значения функции

$$y = \begin{cases} |1 - x|, & \text{при } x < -3 \\ -x, & \text{при } -1 \leq x < 0 \\ 1 - x, & \text{при } x \geq 7 \\ \ln|x|, & \text{в остальных случаях} \end{cases}$$

9. Составить алгоритм расчета премии: если оклад сотрудника  $> 5000$  процент премии составляет 10 %; если оклад  $< 5000$ , то процент премии равен 12 %.
10. Составить алгоритм для нахождения суммы двух наибольших чисел из четырех заданных.
11. Составить алгоритм вывода названия дня недели по его порядковому номеру (1 – понедельник, 2 – вторник, 3 – среда, 4 – четверг, 5 – пятница, 6 – суббота, 7 – воскресенье).
12. Дано натуральное число  $n$  и действительное число  $x$ . Вычислить  $S = \sin x + \sin 2x + \dots + \sin nx$ .
13. Дано целое число  $N (> 0)$ . Если  $N$  – нечетное, то вывести произведение  $1 \cdot 3 \cdot \dots \cdot N$ ; если  $N$  – четное, то вывести произведение  $2 \cdot 4 \cdot \dots \cdot N$ .
14. Составить алгоритм вычисления суммы  $S = 2^2 + 2^3 + 2^4 + \dots + 2^{10}$ .

15. Составить алгоритм табулирования функции  $y = \begin{cases} \sin^2(1+x) & \text{при } x \neq 0 \\ x \lg x & \text{при } x = 0 \end{cases}$

на отрезке  $[a;b]$  с шагом  $h = 0,1$ .

16. Известны данные о мощности (в единицах силы) двигателей 30 моделей легковых автомобилей. Составить алгоритм определения среди них количества моделей, мощность двигателя которых превышает 200 единиц силы.

17. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10 % от пробега предыдущего дня. Составить алгоритм определения, в какой день суммарный пробег за все дни превысит 100 км.

18. Составить алгоритм вычисления следующего выражения  $y = \sum_{i=1}^3 i + \prod_{j=1}^5 j^2$ .

19. Группа студентов из 20 человек в сессию сдавала три экзамена. Составить алгоритм заполнения экзаменационной ведомости.

20. Составить алгоритм вычисления следующего выражения  $y = \sum_{i=1}^3 \sum_{j=1}^4 (i + j)$ .

21. Даны две последовательности чисел  $A_i$  ( $i = 1, \dots, 30$ ) и  $C_j$  ( $j = 1, \dots, 25$ ). Найти количество отрицательных элементов в каждой из них. Определить, в какой последовательности больше отрицательных элементов.

22. Составить алгоритм табулирования функции  $y = \begin{cases} x + t^2 \\ x - t^2 \end{cases}$  на отрезке  $[a;b]$  с

шагом  $h$ , если  $t = 2$ ,  $a = -5$ ,  $b = 5$ ,  $n = 10$ .

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Строго определенная последовательность действий, необходимая для решения данной задачи, – это:

- 1) метод решения;
- 2) алгоритм;
- 3) порядок действий.

2. Укажите, какие из перечисленных свойств не относятся к основным свойствам алгоритма:

- 1) дискретность;
- 2) определенность;
- 3) Актуальность;
- 4) результативность;
- 5) массовость;

- б) строгость.
3. Алгоритм, записанный на «понятном» компьютеру языке программирования, называется:
- 1) исполнителем алгоритмов;
  - 2) программой;
  - 3) псевдокодом;
  - 4) протоколом алгоритма.
4. Свойство, состоящее в том, что любой алгоритм должен завершаться за конечное (пусть даже очень большое) число шагов это:
- 1) дискретность;
  - 2) результативность;
  - 3) однозначность.
5. Графический способ – это способ описания алгоритмов:
- 1) с помощью геометрических фигур с линиями связи, показывающими порядок выполнения отдельных инструкций;
  - 2) с помощью графических редакторов;
  - 3) с помощью графических операторов.
6. Псевдокоды – это способ описания алгоритма:
- 1) с помощью слов и формул;
  - 2) с помощью специальных графических схем алгоритмов;
  - 3) с помощью естественного языка;
  - 4) с помощью языка машинных кодов.
7. Свойством алгоритма является:
- 1) результативность;
  - 2) цикличность;
  - 3) возможность изменения последовательности выполнения команд;
  - 4) возможность выполнения алгоритма в обратном порядке;
  - 5) простота записи на языках программирования.
8. Формальное исполнение алгоритма – это:
- 1) исполнение алгоритма конкретным исполнителем с полной записью его рассуждений;



- 2) разбиение алгоритма на конкретное число команд и пошаговое их исполнение;
- 3) исполнение алгоритма не требует рассуждений, а осуществляется исполнителем автоматически;
- 4) исполнение алгоритма осуществляется исполнителем на уровне его знаний.

9. Алгоритмом является:

- 1) студенческий билет;
- 2) правила поведения в вузе;
- 3) номер группы;
- 4) схема расположения аудитории.

10. Расставьте действия в нужном порядке – Алгоритм «Выполнение аппликации»:

- 1) положить клей, ножницы и бумагу на место;
- 2) вырезать фигуры нужного цвета;
- 3) взять трафарет;
- 4) намазать клеем вырезанные фигуры;
- 5) приклеить аппликацию;
- 6) взять клей и ножницы;
- 7) подобрать подходящую по цвету бумагу и трафарет.

11. Выберите исполнителей алгоритма:

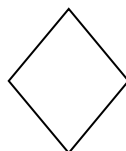
- 1) Робот;
- 2) человек;
- 3) компьютер;
- 4) бытовой прибор.

12. В блок-схеме условие обозначается блоком:

а)



б)



в)



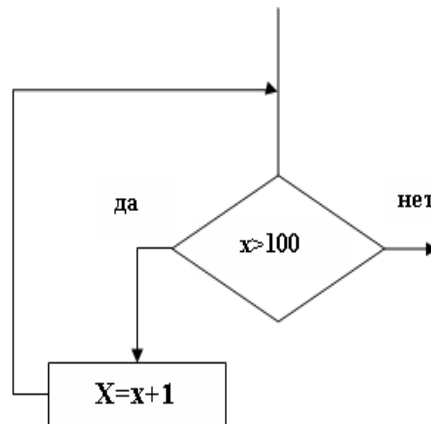
13. Представленная фигура в блок-схеме обозначает:

- 1) ввод–вывод данных;
- 2) вычислительный процесс;
- 3) начало/конец алгоритма.



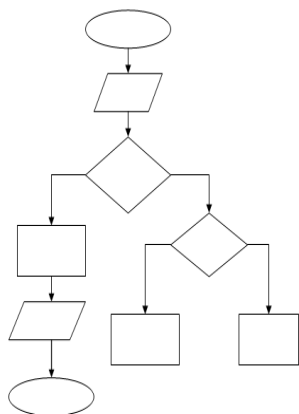
14. На рисунке приведена схема:

- 1) линейной конструкции;
- 2) конструкции неполного ветвления;
- 3) конструкции полного ветвления.

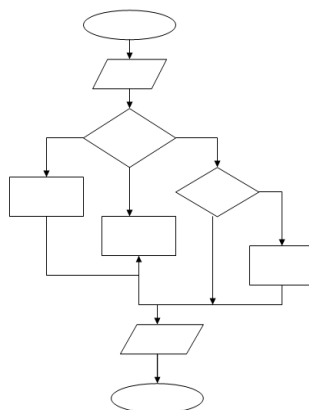


15. Из нижеприведенных блок-схем укажите правильную:

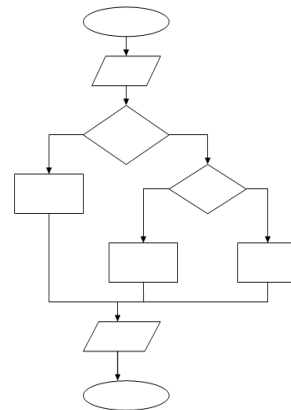
а)



б)



в)

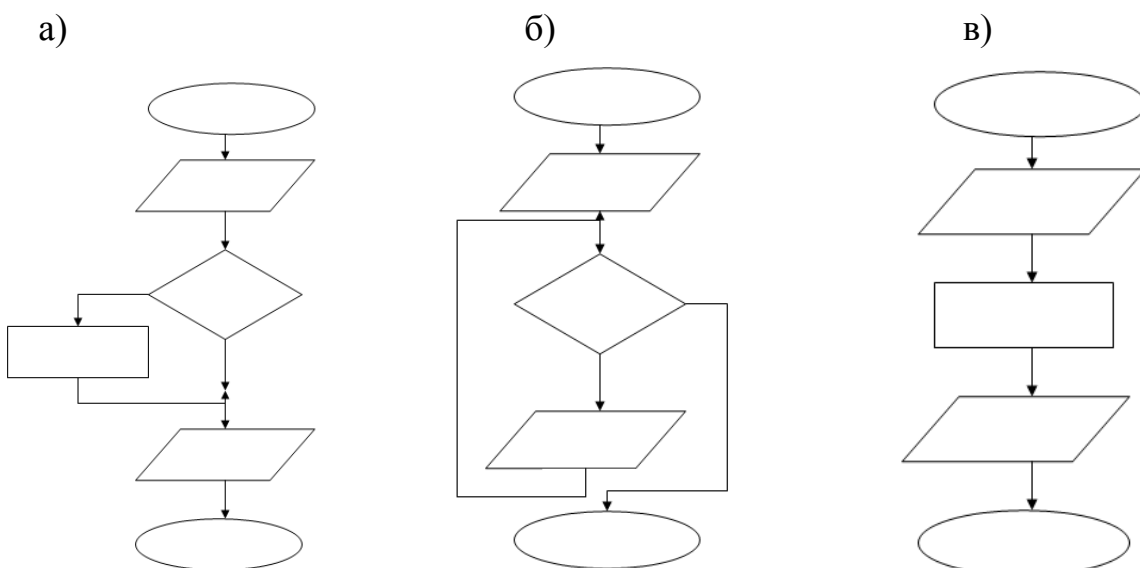


16. Укажите, какие из перечисленных конструкций не относятся к основным группам алгоритмов:

- 1) линейные;
- 2) разветвляющиеся;
- 3) вложенные;
- 4) циклические.

17. Алгоритм, записанный на специальном языке, понятном компьютеру, на языке программирования, называется:

- 1) исполнителем;
  - 2) программой;
  - 3) системой команд исполнителя;
  - 4) блок-схемой.
18. Алгоритм, в котором все этапы решения задачи выполняются строго последовательно, называется:
- 1) линейным;
  - 2) разветвляющимся;
  - 3) циклическим.
19. Геометрическая фигура ромб используется в блок-схемах для обозначения:
- 1) начала и конца алгоритма;
  - 2) ввода или вывода данных;
  - 3) условия;
  - 4) действия.
20. Рекурсивным алгоритмом называется:
- 1) алгоритм, который в процессе выполнения на каком-либо шаге прямо или косвенно обращается сам к себе;
  - 2) алгоритм, в котором несколько раз повторяется одно действие;
  - 3) алгоритм, в котором реализуется операция «выбор».
21. Какая из приведенных блок-схем является блок-схемой алгоритма линейной структуры?

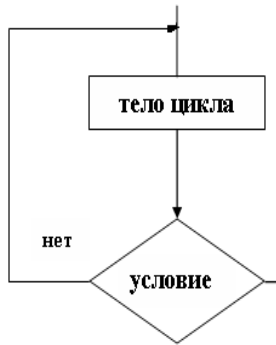


22. Цикл, в котором количество повторений заранее определено, называется:

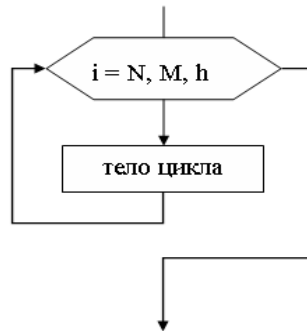
- 1) арифметическим;
- 2) циклом с постусловием;
- 3) циклом с предусловием.

23. Какая из приведенных блок-схем является блок-схемой цикла с постусловием:

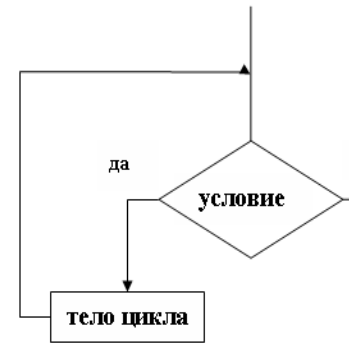
а)



б)



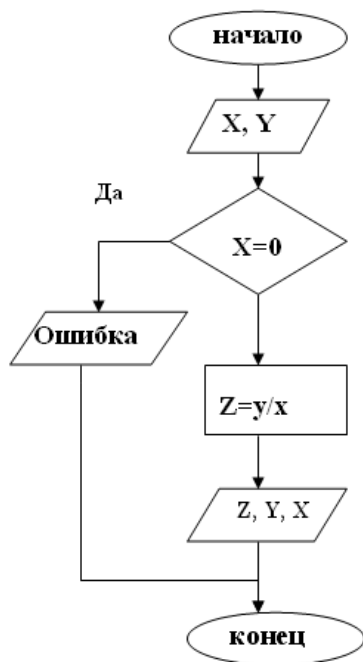
в)



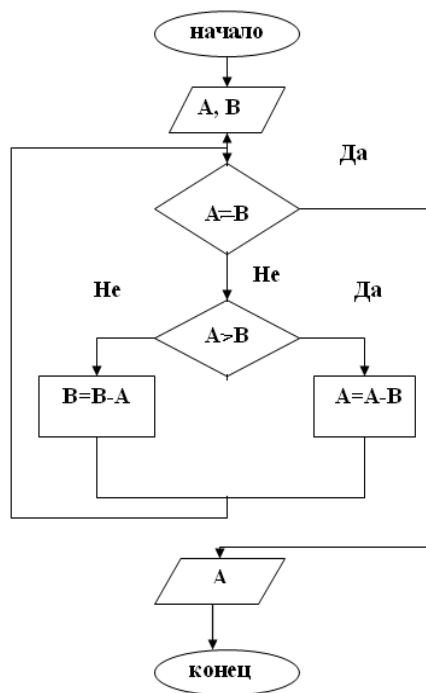
24. Расставьте этапы решения задачи последовательно:

- 1) тестирование программы;
- 2) выбор метода решения;
- 3) составление и отладка программы;
- 4) решение поставленной задачи и представление результатов;
- 5) математическая формулировка задачи;
- 6) составление алгоритма решения;
- 7) общая формулировка задачи.

25. Что получится на выходе блок-схемы, если  $x = 1$ ,  $y = 0$ ?



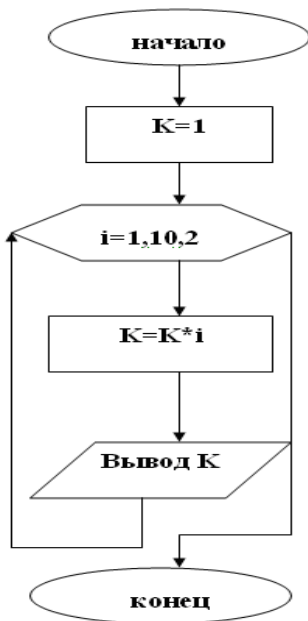
26. Найти A на выходе блок-схемы, если A = 5, B = 10



27. Циклы, для которых число повторений определяется в ходе выполнения цикла, называются:

- 1) итерационными;
- 2) арифметическими;
- 3) вложенными.

28. Какое значение примет переменная K после выполнения алгоритма?



29. Выберите один из вариантов ответа на вопрос: «Когда окончится выполнение цикла?»:

**начало цикла**

**пока**  $a < b$

**делать**

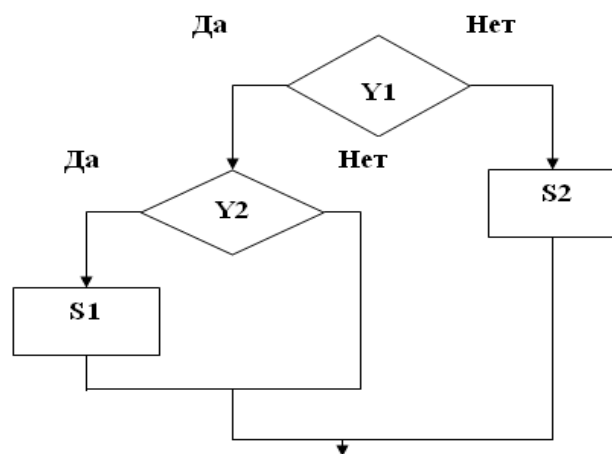
$a = a + 1$

**конец цикла**

- 1) когда  $a$  станет больше  $b$ ;
- 2) когда  $a$  станет равно  $b$ ;
- 3) цикл не закончится;
- 4) сразу закончится.

30. Выбрать условие, при котором будет выполняться команда S1:

- 1)  $(Y1 = \text{истина}) \text{ И } (Y2 = \text{истина})$ ;
- 2)  $(Y1 = \text{истина}) \text{ ИЛИ } (Y2 = \text{ложь})$ ;
- 3)  $(Y1 = \text{истина}) \text{ И } (Y2 = \text{ложь})$ .



31. Комбинация двух циклов, изменяющихся одновременно, называется:
- 1) вложенные циклы;
  - 2) последовательные циклы;
  - 3) запрещенные циклы.
32. Задача определения количества мальчиков в группе детского сада использует алгоритм:
- 1) накопления суммы;
  - 2) накопления произведения;
  - 3) подсчета количества элементов.

## ЗАКЛЮЧЕНИЕ

Быстрое развитие информационных технологий постоянно заставляет вносить изменения в содержание курса «Информатика». Некоторые темы дисциплины изменяются явно, многие другие по своему существу остаются без изменений. Алгоритмизация, как раздел информатики, который изучает процессы создания алгоритмов, традиционно относится к теоретической информатике вследствие своего фундаментального характера.

Развитие новых информационных технологий, и в частности технологий программирования, делает возможным в пределах раздела «Основы алгоритмизации» развивать умения и навыки, необходимые пользователю при работе с современным программным обеспечением, т. е. появляется возможность сделать этот раздел связующим звеном между теоретической и практической информатикой.

Проектирование любой задачи, вне зависимости от уровня ее сложности и сферы применения, ставит разработчика перед необходимостью выбора способа ее решения и составления алгоритма.

В учебном пособии сделана попытка систематизировать информацию по алгоритмизации, рассматриваются приемы и методы построения алгоритмов, излагаются практически все основные алгоритмические конструкции, для многих из них приводятся примеры. Данное издание предлагает инструмент для применения основ алгоритмизации и программирования в практической деятельности.

Изучение основ алгоритмизации должно быть ориентированным именно на развитие алгоритмического мышления, а не на изучение определенного языка программирования. Целью изучения данной темы является научить студентов анализировать и ставить задачи, разрабатывать алгоритмы их решения.



## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Аляев Ю. А. Алгоритмизация и языки программирования Pascal, C++, Visual Basic : учеб.-справ. пособие для вузов / Ю. А. Аляев. – М. : Финансы и статистика, 2004. – 320 с.
2. Аляев Ю. А. Практикум по алгоритмизации и программированию на языке Паскаль : учеб. пособие / Ю. А. Аляев, В. П. Гладков, О. А. Козлов. – М. : Финансы и статистика, 2004. – 528 с.
3. Златопольский Д. М. Сборник задач по программированию / Д. М. Златопольский. – 2-е изд., перераб и доп. – СПб. : БХВ-Петербург, 2007. – 240 с.
4. Информатика : учебник / Б. В. Соболев [и др.]. – Изд. 3-е, дополн. и перераб. – Ростов н/Д : Феникс, 2007. – 446 с.
5. Информатика и информационные технологии / под ред. Ю. Д. Романовой. – М. : Эксмо, 2005. – 592 с.
6. Князева М. Д. Алгоритмика: от алгоритма к программе : учеб. пособие / М. Д. Князева. – М. : КУДИЦ-ОБРАЗ, 2006. – 191 с.
7. Колдаев В. Д. Основы алгоритмизации и программирования : учеб. пособие / В. Д. Колдаев ; под ред. проф. Л. Г. Гагариной. – М. : ФОРУМ: ИНФРА – М, 2006. – 416 с.
8. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М. : МЦНМО, 2002. – 955 с.

# ОГЛАВЛЕНИЕ