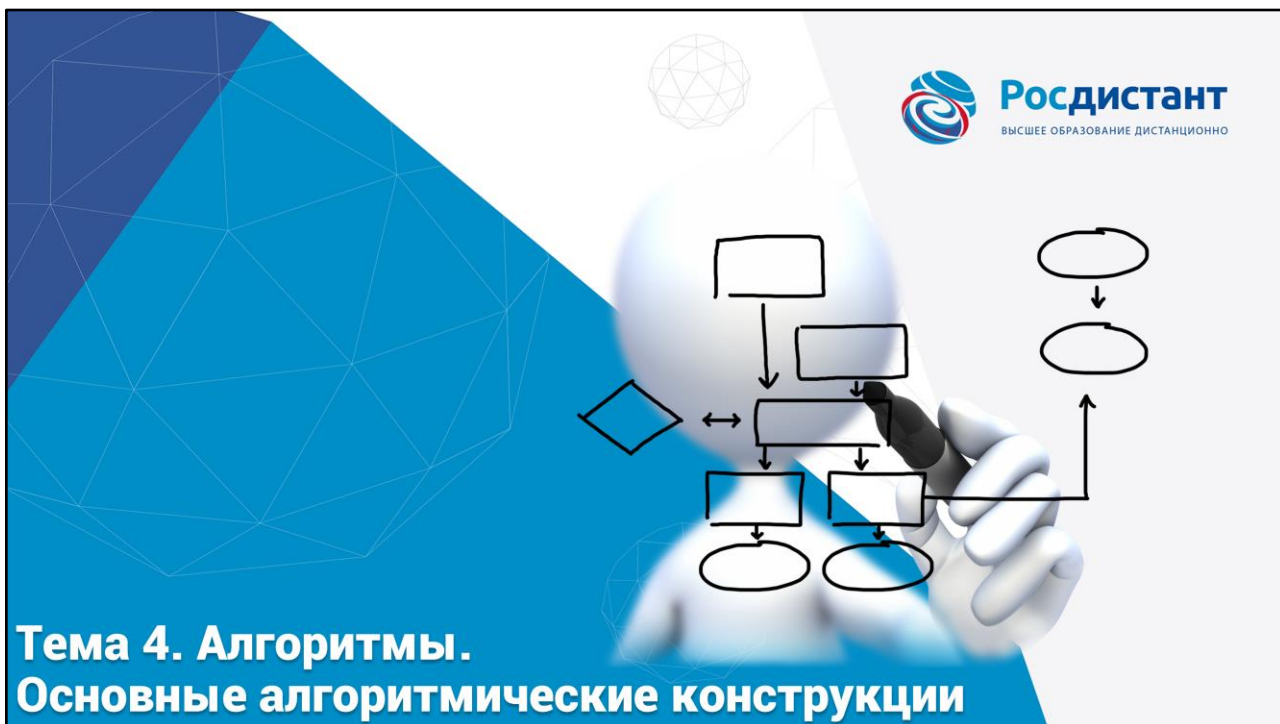




Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

МЕТОДЫ РЕШЕНИЯ ПРОБЛЕМ В

ИНФОРМАТИКЕ 4 ЧАСТЬ



Тема 4. Алгоритмы. Основные алгоритмические конструкции

В рамках данной темы предусмотрено знакомство с основными понятиями алгоритмов, историей их возникновения, видами алгоритмов и классами задач, для которых их применяют. Будут изучены основные свойства алгоритмов и основные формы их представления, плюсы и минусы использования разных форм представления алгоритмов. Особое внимание будет уделено основным алгоритмическим структурам. Материал темы обучает различению основных типов алгоритмов в зависимости от их функций и круга решаемых задач.

АЛГОРИТМ

Основная цель алгоритма - получить конкретный результат

Алгоритм состоит из нескольких непрерывных шагов

Результат приходит после того, как алгоритм завершил весь процесс

Алгоритм - «последовательность шагов, которые необходимо выполнить для получения требуемого выхода из определенного входа»



В самом чистом смысле алгоритм – это математический процесс для решения проблемы с использованием конечного числа шагов.

Алгоритм можно определить как «последовательность шагов, которые необходимо выполнить для получения требуемого выхода из определенного входа». Из этого определения можно выделить 3 основные особенности алгоритма.

Первое. Основная цель алгоритма – получить конкретный результат.

Второе. Алгоритм состоит из нескольких непрерывных шагов.

Третье. Результат приходит после того, как алгоритм завершил весь процесс.

По сути, все алгоритмы работают логически, следуя шагам до получения результата для заданного входа.

В мире компьютеров алгоритм – это набор инструкций, который определяет не только то, что нужно сделать, но и то, как это делать.

Вероятно, лучший способ понять алгоритм – это думать о нем как о рецепте.

Есть много способов испечь печенье, но, следуя рецепту, который знает пекарь, нужно сначала предварительно разогреть духовку, затем отмерить муку, добавить масло, шоколадную стружку и т. д., пока желаемое печенье не будет готово.

Используя алгоритмы, программист или ученый-компьютерщик может:

- приказать своей машине запросить в базе данных данные о продажах за последний месяц;
- сравнить их с данными предыдущего месяца и того же месяца прошлого года;

- отобразить это на гистограмме.

Соедините несколько алгоритмов вместе, и вы получите работающую компьютерную программу.

ИСТОРИЯ АЛГОРИТМА

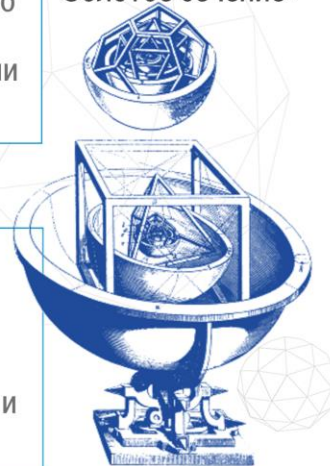
Алгоритмы связаны с программным обеспечением и компьютерами

Упоминание в книгах по искусству эпохи Возрождения, в истории музыки

«Алгоритм - это инструкция о том, как действовать, которая состоит из конечного числа правил»

«Алгоритм - это ограниченная последовательность однозначных элементарных инструкций, которые точно и полностью описать способ решения проблемы»

«Ф – Число Бога.
Золотое сечение»



 **Росдистант**
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

В наши дни алгоритмы в основном связаны с программным обеспечением и компьютерами, но их истоки уходят далеко в прошлое. Алгоритмы использовались интуитивно на протяжении веков, например, в форме систем регулирования, инструкций, правил для игр, архитектурных планов и музыкальных партитур.

Даже некоторые иллюстрированные книги по искусству эпохи Возрождения, например «Четыре книги об измерениях» Альбрехта Дюрера 1525 года, на самом деле были структурированными инструкциями по созданию картин, скульптур и зданий. В истории музыки, от Баха и Моцарта до Шёнберга и Шиллингера, мы видим, что математические методы и даже небольшие механические устройства используются для облегчения процесса музыкальной композиции.

Что именно подразумевается под термином «алгоритм»?

«Алгоритм – это процедура для принятия решения или инструкция о том, как действовать, которая состоит из конечного числа правил».

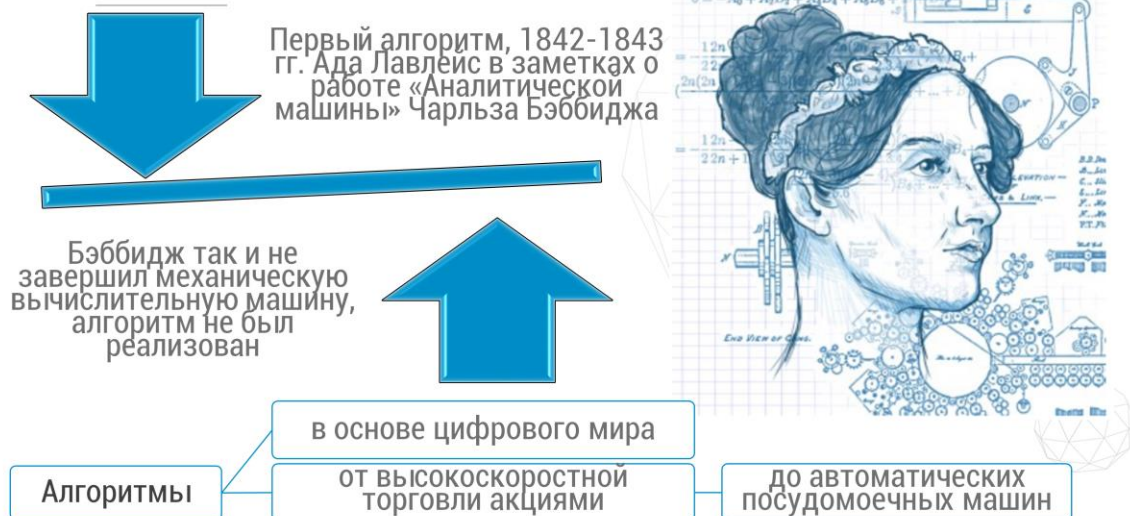
Или: «... ограниченная последовательность однозначных элементарных инструкций, которые точно и полностью описывают способ решения конкретной проблемы».

Эти определения равно применимы к математике, изобразительному искусству или музыке.

Тем не менее наиболее известным применением алгоритмов, несомненно, является их использование в компьютерном программировании. Программа –

это алгоритм, сформулированный на языке, который позволяет обрабатывать ее на компьютере. Следовательно, каждая компьютерная программа – более сложный машинный язык – представляет собой алгоритм. Таким образом, люди передают работу по обработке процедур, необходимых для производства или принятия решений, часто – вычислений, требующих дней или часов, на машину.

ИСТОРИЯ АЛГОРИТМА



Первый алгоритм, разработанный для механической вычислительной машины, был написан в 1842–1843 годах британским математиком Адой Лавлейс. Она описала его в своих заметках о работе «Аналитической машины» Чарльза Бэббиджа, разработанной в 1833 году. Алгоритм предназначался для вычисления вероятностей с использованием чисел Бернулли.

Однако, поскольку английский изобретатель, математик и философ так и не сумел завершить свою механическую вычислительную машину для общих приложений при жизни, алгоритм для нее также не был реализован. «Машину Бэббиджа так и не построили, потому что она была слишком громоздкой и сложной, даже несмотря на то, что программное обеспечение для нее было разработано». Это сообщает доктор Мануэль Бахманн в своей книге «Торжество алгоритма. Как возникла идея программного обеспечения». Тем не менее свет увидели другие вычислительные машины, и параллельно возникла необходимость в более точном программировании.

Алгоритмы лежат в основе практически всего в цифровом мире, от высокоскоростной торговли акциями до автоматических посудомоечных машин. Поскольку технологии становятся все более распространенными, мы полагаемся на умные автомобили, умные дома, умные города. Нам может показаться, что мы взаимодействуем с совершенно новой формой сознания на планете, которая ходит, разговаривает и думает.

На самом деле, это просто множество чисел, обрабатываемых множеством алгоритмов.

ИСТОРИЯ АЛГОРИТМА

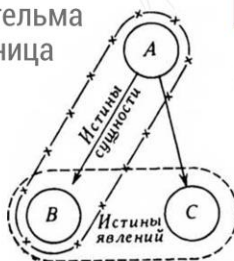


разработал концепцию АЛГОРИТМА в математике, что является причиной того, что его называют дедушкой компьютерных наук

Сон Пифагора



Видение Готфрида Вильгельма Лейбница



Новый вид науки Стивена Вольфрама



Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

В последние десятилетия алгоритмы стали ключевым аспектом информатики и, в частности, теории сложности и вычислимости. Любую проблему, которую можно запрограммировать, можно решить с помощью алгоритма.

Даже самые сложные вещи можно вычислить с помощью единиц и полей. Например, Сон Пифагора – это способность объяснять мир отношениями между целыми числами. А видение Готфрида Вильгельма Лейбница опирается на представление о том, что все рациональные истины основаны на своего рода исчислении. Это отражено в цифровой философии и взгляде на мир, основанном на алгоритмах, описанном в книге «Новый вид науки», написанной в 2002 году британским физиком и математиком Стивеном Вольфрамом. Используя многочисленные наглядные примеры, он описывает способность клеточных автоматов объяснять природу в сравнении с более традиционными математическими моделями. Его взгляды весьма противоречивы в научном сообществе, как это часто случалось в истории алгоритмов.

Слово «алгоритм» происходит от имени математика Мохаммеда ибн-Мусы Аль-Хорезми, который был членом королевского двора в Багдаде и жил примерно с 780 по 850 годы. Работа Аль-Хорезми является вероятным источником алгебры. Аль-Хорезми разработал концепцию алгоритма в математике, что является причиной того, что некоторые люди называют его дедушкой компьютерных наук.

Алгоритм – это процедура или формула для решения проблемы, основанная на выполнении последовательности определенных действий.

Компьютерную программу можно рассматривать как сложный алгоритм. В математике и информатике алгоритм обычно означает небольшую процедуру, которая решает повторяющуюся проблему. Алгоритмы широко используются во всех областях информационных технологий.

ВИДЫ АЛГОРИТМОВ



В любой сфере сегодня применяются алгоритмы. Можно увидеть применение алгоритмов практически для каждого вида математических задач, а также для других областей:

- численные алгоритмы;
- алгебраические алгоритмы;
- геометрические алгоритмы;
- последовательные алгоритмы;
- алгоритмы работы с величинами;
- теоретические алгоритмы.

Существуют также различные алгоритмы, названные в честь ведущих изобретателей-математиков: алгоритм Шора, алгоритм Гирвана – Ньюмана, несколько алгоритмов Евклида.

Есть алгоритмы, названные в честь конкретной проблемы, которую они решают, например алгоритм двунаправленного поиска, К-образный алгоритм слияния. В области вычислений большинство алгоритмов решают задачи управления и анализа данных.

Таким образом, диапазон алгоритмов обширен.

ИСПОЛНИТЕЛИ АЛГОРИТМА

Исполнитель алгоритма – это некоторая абстрактная или реальная система, способная выполнить действия, предписываемые алгоритмом

Исполнителя характеризуют:

- среда
- система команд
- элементарные действия
- отказы



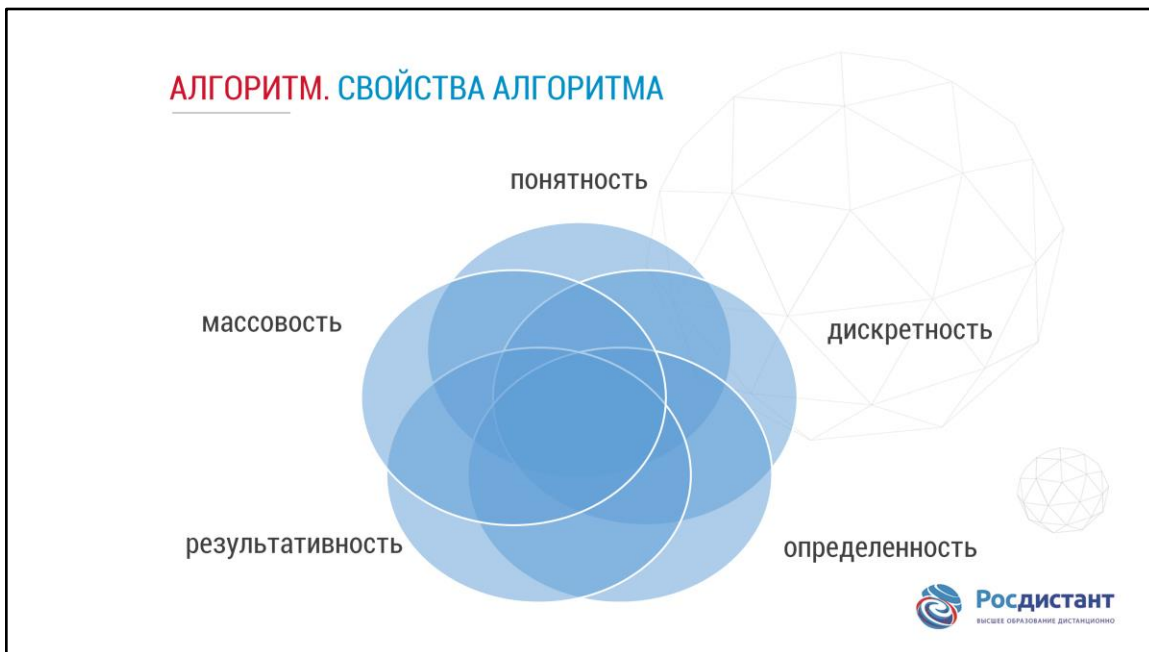
Алгоритм – заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов. Исполнитель алгоритма – это некоторая абстрактная или реальная система, способная выполнить действия, предписываемые алгоритмом.

Исполнителя характеризуют:

- среда – «место обитания» исполнителя;
- система команд. Каждый исполнитель может выполнять команды только из некоторого строго заданного списка – системы команд исполнителя. Для каждой команды должны быть заданы условия применимости, в каких состояниях среды может быть выполнена команда, и описаны результаты выполнения команды;
- элементарные действия. После вызова команды исполнитель совершает соответствующее элементарное действие;
- отказы. Отказы исполнителя возникают, если команда вызывается при недопустимом для нее состоянии среды.

Обычно исполнитель ничего не знает о цели алгоритма. Он выполняет все полученные команды, не задавая вопросов «почему» и «зачем».

В информатике универсальным исполнителем алгоритмов является компьютер.



Эффективность работы алгоритма и правильность полученных результатов обеспечиваются следующими свойствами алгоритма.

Понятность для исполнителя. Исполнитель алгоритма должен понимать, как его выполнять. Имея алгоритм и произвольный вариант исходных данных, исполнитель должен знать, как надо действовать для выполнения этого алгоритма.

Дискретность, прерывность, раздельность. Алгоритм должен представлять собой процесс решения задачи как последовательное выполнение простых или ранее определенных шагов, этапов.

Определенность. Каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвольного понимания. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

Результативность, или конечность, состоит в том, что за конечное число шагов алгоритм:

- либо должен приводить к решению задачи;
- либо после конечного числа шагов останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения;
- либо неограниченно продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов.

Массовость означает, что алгоритм решения задачи разрабатывается в общем виде, т. е. он должен быть применим для некоторого класса задач,

различающихся лишь исходными данными.

Если разработанный алгоритм не удовлетворяет какому-либо из названных свойств, то в дальнейшем это приведет к нежелательным последствиям, которые могут быть выявлены на одном из следующих этапов работы с задачей. Так, невыполнение свойства массовости алгоритма приведет к тому, что он окажется применимым только для обработки одного заданного набора исходных данных, а для других данных придется разрабатывать новый алгоритм.



Используются следующие формы представления алгоритмов:

- словесная – запись алгоритма на естественном разговорном языке;
- графическая – использование для представления алгоритма графических фигур;
- псевдокод – описание алгоритма на условном алгоритмическом языке, включающем как элементы, характерные для языка программирования, так и фразы разговорного языка, а также общепринятые математические и иные обозначения;
- программа – запись алгоритма на одном из языков программирования.

Словесный способ записи алгоритма представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.

Словесный способ не имеет широкого распространения, так как такие описания:

- строго не формализуемы;
- страдают многословностью;
- допускают неоднозначность толкования отдельных предписаний.

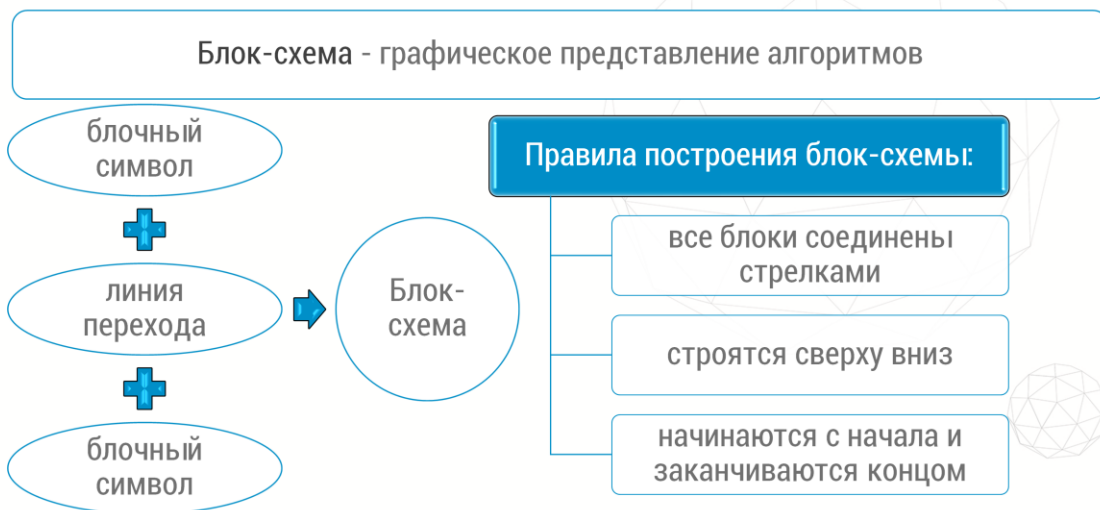
Графический способ записи алгоритма представляет собой изображение в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Программный способ предполагает перенос алгоритма на язык программирования.

Наибольшее распространение в информатике получили блок-схемы и

программный способ представления алгоритма.

БЛОК-СХЕМА









Чаще других используется графическое представление алгоритмов, при котором алгоритм изображается в виде последовательности связанных между собой геометрических фигур, называемых функциональными блоками. Такое графическое представление называется схемой алгоритма, или блок-схемой. Блок-схемы – это схемы, которые наглядно представляют процесс решения проблем.

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, представленная в виде **блочного символа**. Блочные символы соединяются **линиями переходов**, определяющими очередность выполнения действий. Умение использовать их при рисовании блок-схем имеет решающее значение. Вот несколько правил, которые следует знать:

- все блоки блок-схем соединены стрелками, чтобы показать логическую связь между ними;
- блок-схемы будут проходить сверху вниз;
- все блок-схемы начинаются с начального поля и заканчиваются клеммной коробкой.

С помощью алгоритмов мы легко можем понять программу. Основная цель блок-схемы – анализировать различные процессы.

БЛОЧНЫЕ СИМВОЛЫ

Вид блока	Назначение	Вид блока	Назначение
	Начало, конец алгоритма		Проверка условия
	Ввод данных		Организация цикла
	Присваивание значений		Вывод результатов

В блок-схеме каждому типу действий, например вводу исходных данных, выводу результата или проверке некоторого условия, соответствует блок определенной формы. Форма блока однозначно определяет вид выполняемого действия. Блоки соединяются линиями переходов, определяющими очередность выполнения действий. Блок-схема выстраивается в направлении либо сверху вниз, либо слева направо.

Блоки «Начало» и «Конец» используются для обозначения начала или конца алгоритма.

Блок «Ввод данных» предполагает ручной ввод значений переменных. Имена переменных, которые необходимо ввести, указываются внутри этого блока.

Блок «Присваивание значений» используется для вычисления значений по формулам и присваивания вычисленных значений переменным.

Блок «Проверка условия» предполагает выбор одного из двух альтернативных путей работы алгоритма в зависимости от выполнения или невыполнения проверяемого условия. При этом условие записывается внутри блока.

Блок «Организация цикла» обеспечивает многократное выполнение некоторой последовательности действий, которая называется «телом цикла». Этот блок применяется в тех алгоритмах, где количество повторений цикла можно определить заранее. Внутри блока указывается переменная, являющаяся параметром цикла, задаются ее начальное значение, конечное значение и шаг изменения.

Блок «Вывод результатов» предполагает вывод значений переменных. Имена

переменных, значения которых необходимо вывести, указываются внутри блока.

АЛГОРИТМЫ И БЛОК-СХЕМЫ



Алгоритм	Блок-схема
Это процедура решения проблем	Это графическое изображение процесса
Процесс показан в пошаговой инструкции	Процесс показан на блок-схеме информационной схемы
Это сложный процесс и сложен в понимании	Это интуитивно понятно и легко
Отлаживать ошибки удобно	Ошибки отлаживать сложно
Решение представлено на естественном языке	Решение представлено в графическом формате
Сложную задачу решить несколько проще	Сложно решить сложную проблему
На создание алгоритма уходит больше времени	На создание блок-схемы уходит меньше времени



Алгоритмы и блок-схемы – это два разных инструмента, используемых для создания новых программ, особенно в компьютерном программировании. Алгоритм – это пошаговый анализ процесса, а блок-схема объясняет шаги программы в графическом виде.

Если сравнить блок-схему с фильмом, то алгоритм – это история этого фильма.

Другими словами, алгоритм – это ядро блок-схемы. На самом деле в области компьютерного программирования существует много различий между алгоритмом и блок-схемой в отношении различных аспектов, таких как точность, способ их отображения и то, как люди относятся к ним.

В таблице подробно показаны различия между ними.

Если алгоритм – это процедура решения проблем, то блок-схема – это графическое изображение процесса. В алгоритме процесс показан в пошаговой инструкции, а в блок-схеме – в самом графическом представлении. Алгоритм очень сложно понять и не всегда сразу получается представить, блок-схема всегда интуитивно понятна и легко воспроизводима. Но в алгоритме удобно отлаживать ошибки, а блок-схема для этого не предназначена. В алгоритме решение представлено на естественном языке, блок-схема предоставляет графическое описание проблемы. С алгоритмом сложную задачу решить несколько проще, в отличие от блок-схемы. На создание алгоритма уходит больше времени, чем на создание блок-схемы.

Алгоритмы в основном используются для математических и компьютерных программ, в то время как блок-схемы могут использоваться для описания всех

видов процессов: деловых, образовательных, личных и других. Таким образом, блок-схемы часто используются в качестве инструмента планирования программы для визуальной организации пошагового процесса выполнения программы.

Из вышесказанного мы можем прийти к выводу, что блок-схема – это наглядное представление алгоритма, алгоритм может быть выражен и проанализирован с помощью блок-схемы.

ПСЕВДОКОД

алг	сим	дано	для	да	цел	кц
арг	лит	надо	от	нет	все	или
рез	лог	если	до	при	вывод	вещ
нач	таб	то	знач	выбор	длин	пока
кон	нц	иначе	и	ввод	не	утв

алг название алгоритма (аргументы и результаты)
дано условия применимости алгоритма
надо цель выполнения алгоритма
нач описание промежуточных величин
последовательность команд (тело алгоритма)
кон



Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов.

В псевдокоде имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. В псевдокоде есть служебные слова, выделяемые в печатном тексте жирным шрифтом, а в рукописном тексте – подчеркиванием. Примером псевдокода является алгоритмический язык, который имеет основные служебные слова, представленные в таблице.

Применяя выделенные служебные слова, алгоритм может быть представлен в обобщенном виде.

Часть алгоритма от слова «алг» до слова «нач» называется заголовком, а часть, заключенная между словами «нач» и «кон», – телом алгоритма.

В предложении «алг» после названия алгоритма в круглых скобках указываются характеристики, включающие перечисление аргументов и результаты. Здесь же описывается тип значения – целый, вещественный, символьный и другие – всех входных аргументов и выходных результатов.

При работе с алгоритмическим языком наиболее широко используются следующие команды:

- команда присваивания – служит для вычисления выражений и присваивания их значений переменным;
- команды ввода и вывода. Ввод – имена переменных; вывод – имена переменных, выражения, тексты;

- команды «если» и «выбор», применяют для организации ветвлений;
- команды «для» и «пока», применяют для организации циклов.

АЛГОРИТМИЧЕСКАЯ СТРУКТУРА «СЛЕДОВАНИЕ»



Алгоритмический язык

действия 1
действия 2
.....
действия N

Построению последовательности действий решения задачи предшествует определение типа вычислительного процесса. Выделяют линейные, разветвляющиеся и циклические алгоритмические структуры, которые можно считать базовыми. Для описания базовых элементов используются язык блок-схем и алгоритмический язык.

Основной характеристикой базовых алгоритмических структур является наличие в них одного входа и одного выхода.

Алгоритмы можно разделить на 3 типа в зависимости от их структуры.

Последовательность. Этот тип алгоритма характеризуется серией шагов, шаги выполняются один за другим.

Ветвление. Этот тип алгоритма представлен проблемами, которые можно выразить схемой «если – то». Если условие истинно, на выходе будет А, если условие ложно, на выходе будет В. Этот тип алгоритма также известен как «тип выбора».

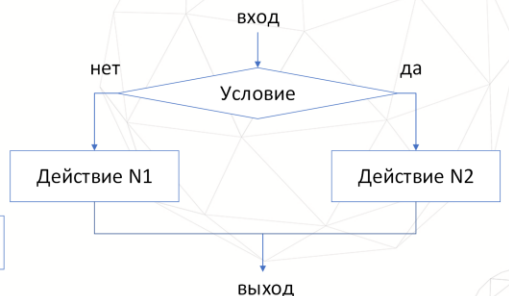
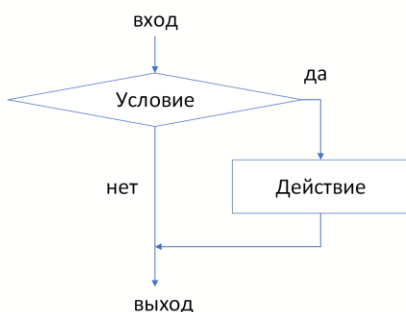
Цикл. Для этого типа процесс может выполняться повторно при определенных условиях. Он представлен «пока» и «для» задач. Необходимо убедиться, что процесс завершится после нескольких циклов при заданном условии. Этот тип алгоритма также известен как «тип повторения».

Алгоритмическая структура, в которой естественный ход выполнения действий не нарушается и действия выполняются строго в том порядке, как они записаны, называется следованием. На слайде представлена алгоритмическая структура «следование».

Под действиями предполагается применение таких операций, как ввод значений исходных данных, присваивание переменным каких-либо значений, вычисление значений переменных по заданным выражениям, вывод полученных результатов.

АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ «ВЕТВЛЕНИЕ»

Если-То



Если-То-Иначе



В алгоритмах, содержащих базовую структуру ветвление, естественный ход вычислений может быть нарушен в зависимости от проверки какого-либо условия.

Выбирается один из возможных путей прохождения алгоритма. Независимо от того, какой путь будет выбран, выполнение алгоритма будет продолжаться, но всегда может быть выбран только один из возможных путей. Это обеспечивает свойство «однозначности» алгоритма.

Разветвляющиеся алгоритмы существуют в двух основных вариантах.

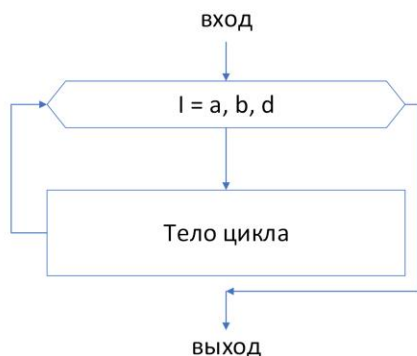
Алгоритмические структуры типа «ветвление» показаны на слайде.

При использовании конфигурации «если – то» проверяется «условие», и если оно выполняется, то выполняется указанное «действие». Если условие не выполняется, «действие» игнорируется. Такая структура называется неполной развилкой. Блок «действие» может содержать любую последовательность операций.

При использовании конфигурации «если – то – иначе» также проверяется «условие», и если оно выполняется, то выполняется «действие 1», иначе выполняется «действие 2». Такая структура называется полной развилкой. Выбор альтернативного пути дальнейшего прохождения алгоритма осуществляется указанием на линиях выхода из блока проверки условия слов «да» или «нет».

АЛГОРИТМИЧЕСКАЯ СТРУКТУРА «ЦИКЛ С ПАРАМЕТРОМ»

Цикл с параметром



Алгоритмический язык

нц для I от a до b шаг d
тело цикла
(последовательность действий)
кц



Базовая алгоритмическая структура цикл предполагает многократное выполнение некоторой совокупности действий.

Различают циклические структуры трех типов:

- цикл с параметром;
- цикл с предусловием;
- цикл с постусловием.

Для организации любого цикла необходимо выполнить три действия:

- определить начальное значение параметра цикла;
- сформировать текущее значение параметра цикла;
- осуществить проверку на выход из цикла.

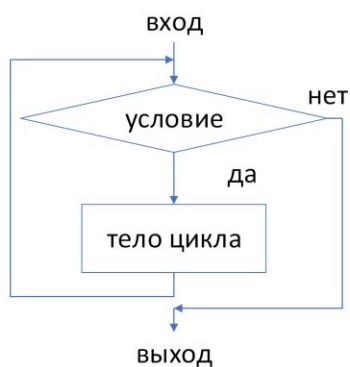
В алгоритмической структуре цикл с параметром, представленной на слайде, указанные действия выполняются в блоке организации цикла.

Здесь I – параметр цикла, a – начальное значение параметра цикла, b – конечное значение параметра цикла, d – шаг изменения параметра цикла.

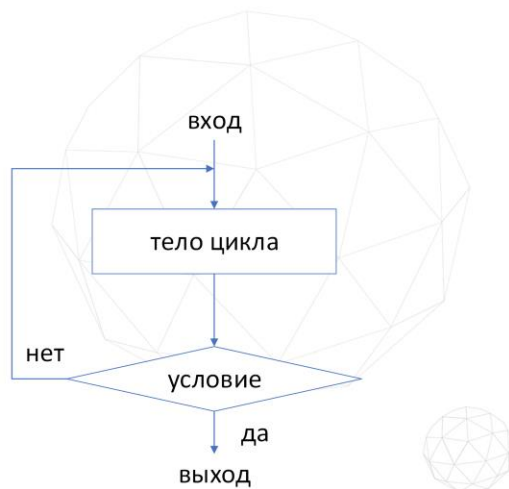
Действия, которые повторяются многократно, образуют «тело цикла». «Цикл с параметром» используют в том случае, когда количество повторений «тела цикла» заранее известно.

Так же представлена запись данной структуры на алгоритмическом языке, которая записывает структурные элементы с использованием служебных слов НЦ и КЦ, что означает «начало цикла» и «конец цикла».

ИТЕРАЦИОННЫЕ ЦИКЛЫ



Цикл с предусловием



Цикл с постусловием

Циклы, для которых число повторений «тела цикла» заранее определить невозможно, называются итерационными. Базовые алгоритмические структуры для организации итерационных циклов существуют в двух вариантах: цикл с предусловием и цикл с постусловием. Выполнение некоторого условия определяет прекращение повторений «тела цикла».

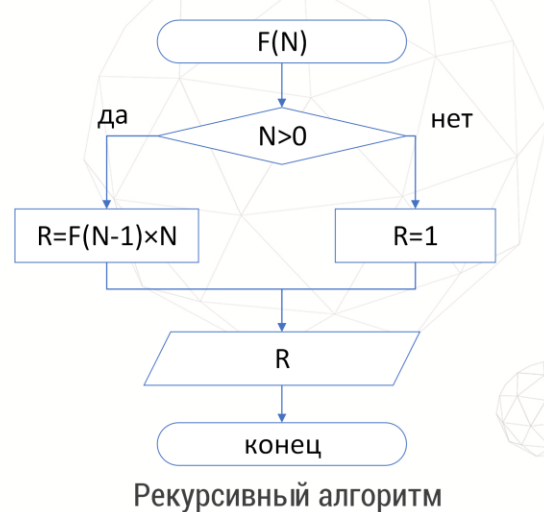
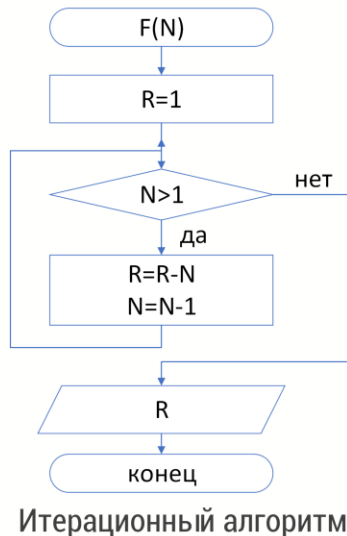
В вычислительных алгоритмах «цикл с предусловием» выполнению «тела цикла» предшествует проверка условия продолжения цикла. Ряд действий, образующих «тело цикла», повторяется до тех пор, пока указанное условие выполняется. Если условие не выполнилось с первого раза, цикл не будет пройден ни разу.

В вычислительных алгоритмах «цикла с постусловием» сначала выполняется ряд действий, образующих «тело цикла», а затем проверяется условие продолжения цикла. Цикл повторяется до тех пор, пока не выполнится указанное условие. Если условие не выполнилось с первого раза, цикл будет пройден один раз.

Алгоритмические структуры «цикл с предусловием» и «цикл с постусловием» можно использовать и в случае известного заранее числа повторений цикла. «Тело цикла» для всех трёх разновидностей алгоритмических структур типа «цикл» может содержать внутри себя любую последовательность действий, включая структуру «ветвление» и даже «цикл».

Мы рассмотрели средства представления алгоритмов с помощью блок-схем. Далее будут показаны примеры составления блок-схем алгоритмов для решения некоторых типовых вычислительных задач.

ИТЕРАЦИОННЫЕ И РЕКУРСИВНЫЕ АЛГОРИТМЫ



Итерация и рекурсия – ключевые методы информатики, используемые при создании алгоритмов и разработке программного обеспечения.

Проще говоря, итеративная функция – это функция, которая выполняет цикл для повторения некоторой части кода, а рекурсивная функция – это функция, которая снова вызывает себя, чтобы повторить код.

Алгоритм, в состав которого входит цикл, называется итерационным. Алгоритм называется рекурсивным, если обращение к этому алгоритму может производиться из него самого.

В большинстве случаев решение об использовании рекурсии подсказывается характером проблемы. Однако важно понимать, что рекурсивность – это свойство решения проблемы, а не атрибут самой проблемы. Чтобы быть подходящим кандидатом для рекурсивного решения, проблема должна иметь три различных свойства:

- возможность разложить исходную проблему на более простые экземпляры той же проблемы;
- поскольку большая проблема разбивается на последовательно менее сложные, эти подзадачи должны в конечном итоге стать настолько простыми, чтобы их можно было решить без дальнейшего деления;
- после того, как каждая из этих более простых подзадач будет решена, должна появиться возможность комбинировать эти решения для получения решения исходной проблемы.

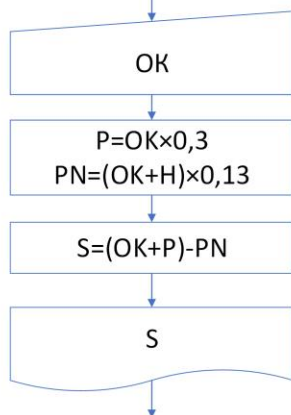
Для проблемы с этими характеристиками рекурсивное решение выполняется

достаточно простым способом. Первый шаг состоит в проверке, подходит ли проблема к категории простых случаев. Если это так, можно решить проблему напрямую. Если нет, нужно разбить всю проблему на новые второстепенные проблемы, имеющие ту же форму, что и исходная проблема.

На рисунках приведены блок-схемы итерационного и рекурсивного алгоритма нахождения факториала.

ПРИМЕР АЛГОРИТМА ЛИНЕЙНОГО ТИПА

Блок-схема:



Пояснения:

- ввод исходных данных
- блок вычислений
- блок вычислений
- вывод результатов



Пример алгоритма линейного типа.

На слайде представлен алгоритм решения следующей задачи.

Ежемесячный размер премии сотрудника предприятия составляет 30 % от оклада. Подоходный налог, взимаемый с сотрудника, составляет 13 %. Требуется вычислить сумму к выдаче сотруднику с учетом премии и подоходного налога. На этапе постановки задачи определяются исходные данные. В данной задаче к исходным данным относится только размер оклада сотрудника. Результатом вычислений будет являться сумма к выдаче.

Используемые обозначения: OK – оклад сотрудника; P – размер премии сотрудника; PN – подоходный налог; S – сумма к выдаче.

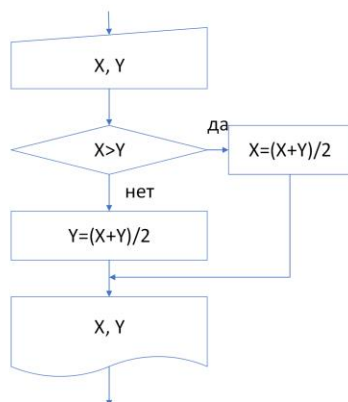
Перед составлением алгоритма необходимо определить тип вычислительного процесса. Анализ алгоритма показывает, что он не содержит структур ветвления или циклов, значит, тип вычислительного процесса и реализующего его алгоритма линейный.

Последовательность действий, описанных в блок-схеме:

- ввод значения переменной OK – размер оклада сотрудника;
- вычисление размера премии сотрудника P ;
- вычисление подоходного налога PN с учетом оклада и премии;
- подсчет суммы к выдаче S , которая складывается из суммы оклада сотрудника и премии, за вычетом подоходного налога;
- вывод результата.

ПРИМЕР АЛГОРИТМА ТИПА «ВЕТВЛЕНИЕ»

Блок-схема:



Пояснения:

- ввод данных
- проверка условия, выбор решения
- вывод результатов



Перейдем к рассмотрению алгоритмов разветвленного типа. Отметим, что каждый такой алгоритм должен содержать как минимум одну базовую алгоритмическую структуру «ветвление».

На слайде представлен алгоритм решения следующей задачи: бóльшее из двух произвольных заданных чисел заменить средним арифметическим значением этих чисел, а в случае их равенства оставить числа без изменения.

Исходными данными для этой задачи являются два произвольных числа, результатом вычислений – преобразованные в соответствии с условием задачи значения этих чисел. В результате выполнения алгоритма большее из двух чисел заменится их средним арифметическим значением, меньшее – не изменится. В случае равенства чисел алгоритм выберет ветвь «нет», второе число заменится средним двух чисел. Так как числа равны, среднее значение будет равно самому числу, следовательно, числа останутся без изменения.

Используемые обозначения: X – первое число; Y – второе число. Перед составлением алгоритма определим тип вычислительного процесса.

Необходимо проверить условие и выбрать один из альтернативных путей решения. Это означает, что алгоритм будет построен с использованием базовой структуры «ветвление».

Последовательность действий, описанных в блок-схеме:

- ввод значений двух переменных;
- проверка условия и выбор дальнейшего пути вычислений: если первое число больше второго, первое заменить средним значением этих чисел, в

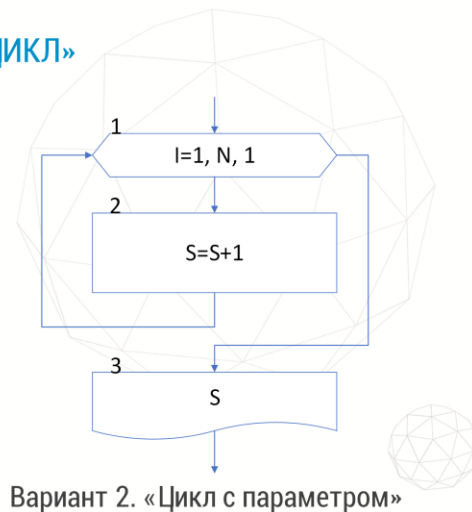
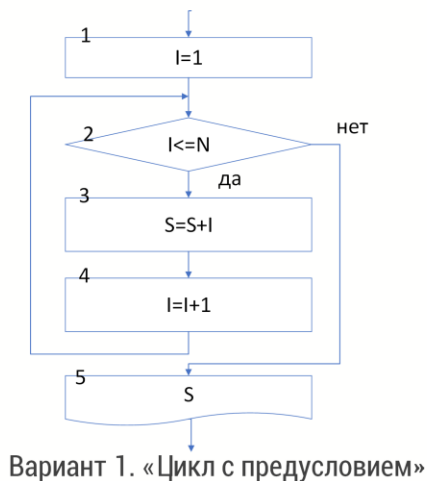
противном случае заменить второе;

- вывод результата.

При любых значениях введенных чисел алгоритм пройдет по одному из возможных путей и приведет к выводу результата.

Далее будут рассмотрены алгоритмы циклического типа, содержащие как минимум одну из базовых структур: «цикл с параметром», «цикл с предусловием» или «цикл с постусловием».

ПРИМЕРЫ АЛГОРИТМОВ ТИПА «ЦИКЛ»



Рассмотрим задачу. Найти сумму первых N натуральных чисел. Напомним, что натуральными называются целые положительные числа 1, 2, 3 и так далее.

На слайде представлены два варианта решения этой задачи. Перечислим используемые обозначения: N – количество натуральных чисел, сумму которых необходимо подсчитать. Значение N должно быть задано; I – параметр цикла; S – сумма чисел.

Вариант 1. Представлена базовая структура «цикл с предусловием». Блоки пронумерованы для удобства описания комментариев к алгоритму:

- блок 1 – определение начального значения параметра цикла;
- блок 2 – проверка условия выхода из цикла;
- блок 3 – вычисление суммы чисел;
- блок 4 – формирование текущего значения параметра цикла;
- блок 5 – вывод результата.

Действия, указанные в блоках 3 и 4, образуют «тело цикла». Действия, указанные в блоках 1, 2 и 4, необходимы для организации цикла. Переменная I является параметром цикла. Ее значение при каждом прохождении цикла увеличивается на единицу. Цикл повторяется до тех пор, пока выполняется условие, указанное в блоке 2.

Вариант 2. «Цикл с параметром». Количество повторений цикла в данной задаче заранее известно, оно равно N . Это позволяет построить алгоритм с использованием базовой структуры «цикл с параметром».

В блоке 1 указан закон изменения параметра цикла I , определены начальное

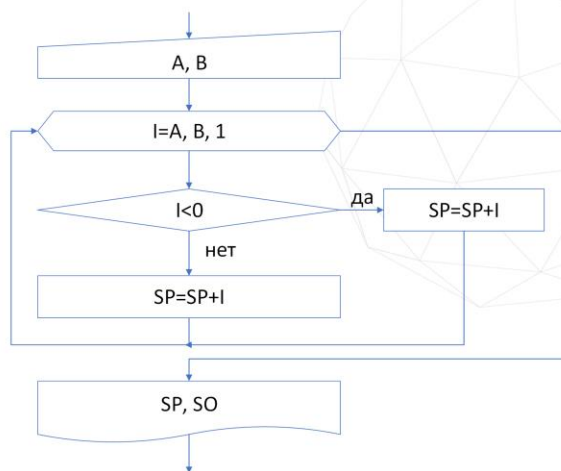
значение, равное единице, конечное значение, равное N , и шаг изменения параметра цикла, равный единице.

В блоке 2 подсчитывается сумма чисел. В блоке 3 выполняется вывод результата.

Действие, указанное в блоке 2, образует «тело цикла».

В блоке 1 объединены действия, указанные в блоках 1, 2 и 4 блок-схемы из варианта 1. Второй вариант блок-схемы более компактен, что наглядно демонстрирует рациональность применения «цикла с параметром» в задачах с известным заранее количеством повторений цикла.

ПРИМЕР АЛГОРИТМА ТИПА «ЦИКЛ»



Рассмотрим задачу, в которой представлено решение с использованием цикла. Задана последовательность целых чисел. Каждое последующее число отличается от предыдущего на единицу. Найти сумму положительных и сумму отрицательных чисел этой последовательности, если известен диапазон изменения этих чисел.

Для решения этой задачи исходными данными являются начало и конец диапазона чисел. Алгоритм задачи должен формировать целые числа из заданного диапазона, при этом алгоритм учитывает, что каждое последующее число вычисляется из предыдущего с помощью добавления к нему единицы. Каждое число анализируется и добавляется либо к сумме положительных, либо к сумме отрицательных чисел.

Используемые обозначения:

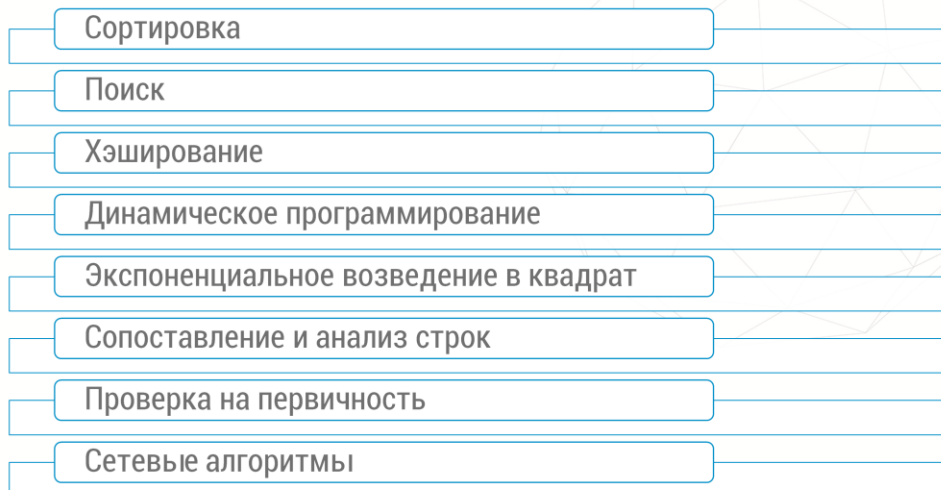
- A – начало диапазона целых чисел, первое число;
- B – конец диапазона целых чисел, последнее число;
- I – число из заданного диапазона;
- SP – сумма положительных чисел;
- SO – сумма отрицательных чисел.

Алгоритм решения задачи показан на слайде. Анализ последовательности действий позволяет определить тип вычислительного процесса как «цикл». Однако в теле цикла необходимо анализировать каждое число, является оно положительным или отрицательным. То есть в «теле цикла» присутствует элемент «ветвления».

Последовательность действий, описанных в блок-схеме, следующая:

- ввод исходных данных – начало и конец диапазона целых чисел;
- построение цикла с использованием блока цикла, в котором указан закон изменения параметра цикла I – от A до B с шагом, равным 1;
- проверка условия – если число отрицательное, то выбирается ветвь «да», в ином случае – ветвь «нет»;
- вывод результатов – сумма положительных чисел SP и сумма отрицательных чисел SO в заданном диапазоне.

ОСНОВНЫЕ ТИПЫ ЗАДАЧ, В КОТОРЫХ ПРИМЕНЯЮТСЯ АЛГОРИТМЫ



Чаще всего алгоритмы применяются для решения следующих типов задач.

Сортировка. Упорядочивание данных эффективным и полезным способом. К таким способам относятся быстрая сортировка, сортировка слиянием, сортировка с подсчетом и другие.

Поиск ключевых данных в отсортированных наборах данных. Чаще всего используется двоичная сортировка, но веб-приложениями также используются поиск по глубине, широте.

Хеширование. Аналогично поиску, но с компонентом индексации и идентификатора ключа. Хеширование обеспечивает превосходные результаты, поскольку присваивает ключ определенным данным.

Динамическое программирование. Преобразует более крупные и сложные проблемы в серии более мелких проблем.

Экспоненциальное возведение в квадрат, также известное как **двоичное возведение в степень**. Данный тип ускоряет вычисление больших целых чисел, многочленов, квадратных матриц и других сложных задач.

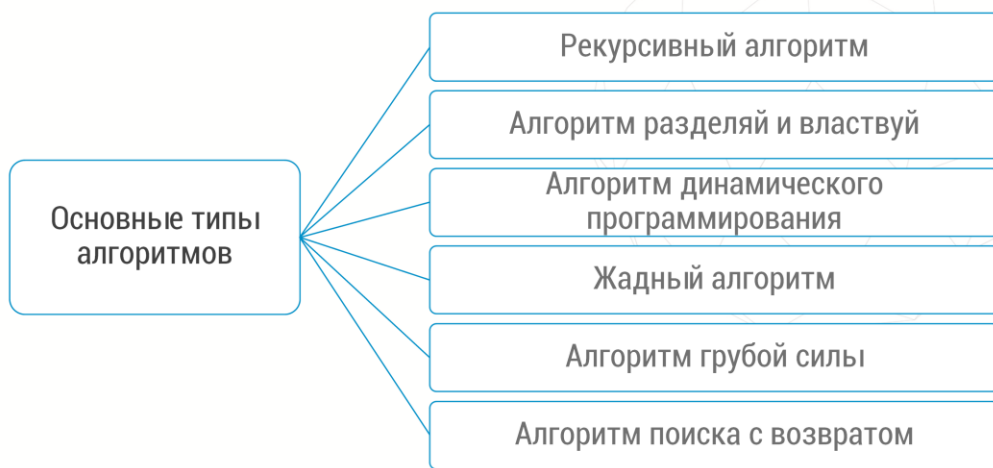
Сопоставление и анализ строк. Предназначен для поиска закономерностей в больших наборах данных с использованием заранее определенных условий и ограничений.

Проверка на первичность. Определяет простые числа детерминированно, или вероятно, в основном используется в криптографии.

Сеть также сильно зависит от алгоритмов, которые управляют всем, от маршрутизации пакетов и управления трафиком до безопасности и

шифрования. Традиционно алгоритмы маршрутизации были в основном статичными, поскольку они устанавливали фиксированные точки сетевой активности.

ОСНОВНЫЕ ТИПЫ АЛГОРИТМОВ В ЗАВИСИМОСТИ ОТ ИХ ФУНКЦИИ



Несмотря на сложность алгоритмов, в целом мы можем разделить алгоритмы на 6 основных типов в зависимости от их функции.

Рекурсивный алгоритм. Это способ решения проблем путем многократного разбиения проблемы на подзадачи одного и того же типа.

Алгоритм «разделяй и властвуй». Традиционно алгоритм «разделяй и властвуй» состоит из двух частей: разбиение проблемы на несколько более мелких независимых подзадач одного типа и нахождение окончательного решения исходных проблем после решения этих более мелких проблем по отдельности.

Алгоритм динамического программирования. В этом типе алгоритма прошлые результаты собираются для использования в будущем. Подобно алгоритму «разделяй и властвуй», алгоритм динамического программирования упрощает сложную задачу, разбивая ее на несколько простых подзадач.

Жадный алгоритм. Заключается в принятии оптимальных решений на каждом этапе, а общее оптимальное решение не рассматривается. Ключ к выбору жадного алгоритма заключается в том, учитывать ли какие-либо последствия в будущем.

Алгоритм грубой силы. Алгоритм грубой силы – это простое и понятное решение проблемы, обычно основанное на описании проблемы и определении задействованной концепции. Алгоритм грубой силы считается одним из простейших алгоритмов, который перебирает все возможности и приводит к удовлетворительному решению.

Алгоритм поиска с возвратом. Основанный на рекурсивном поиске, алгоритм обратного отслеживания фокусируется на поиске решения проблемы во время процесса поиска, подобного перечислению. Когда он не может удовлетворить условию, он вернет «возврат с возвратом» и попробует другой путь. Он подходит для решения больших и сложных задач, за что получил репутацию «общего метода решения».

ТИПЫ АЛГОРИТМОВ



Существует бесконечное количество типов алгоритмов, особенно если учесть тот факт, что любой набор инструкций составляет алгоритм. Перечислим основы наиболее распространенных типов алгоритмов.

Компьютерные алгоритмы связаны с информатикой. Компьютеры работают по алгоритмам для работы, выполняя задачи введением данных – нажатием клавиш или щелчком по определенным кнопкам. Чтобы компьютерная программа работала, программисты создают алгоритмы, которые точно сообщают компьютеру, что делать при каждом конкретном вводе.

Математические алгоритмы широко используются в математических уравнениях. Поскольку математика – это все о наборах правил, входных данных и ответах, то каждая математическая задача, по сути, имеет алгоритм решения.

Алгоритмы социальных сетей. Сайты социальных сетей используют алгоритмы для создания своих страниц, а также для отображения определенной информации определенным пользователям. Эти алгоритмы извлекают данные из пользовательского опыта, чтобы определить, что им интересно.

Алгоритмы и цифровой маркетинг. Поскольку цифровой маркетинг – это онлайн-маркетинг, в основе всего лежат алгоритмы. Интернет, по сути, запрограммирован с помощью множества алгоритмов, и компьютеры тоже работают на них. Алгоритмы социальных сетей также играют важную роль в цифровом маркетинге, поскольку маркетинг в социальных сетях – это огромная и постоянно растущая область маркетинга.

Алгоритмы поисковых систем отображают результаты в ответ на поиск, что

жизненно важно для SEO. Поскольку целью SEO является оптимизация веб-сайтов для поисковых систем, эти алгоритмы влияют на тактику, используемую для повышения рейтинга веб-сайта в релевантных результатах поиска.

ВЫВОДЫ

В компьютерном программировании алгоритмы часто создаются как функции

Программисты стремятся создать максимально эффективные алгоритмы

Используя алгоритмы, разработчики могут обеспечить максимально быструю работу программ и минимальное использование системных ресурсов

Разработчики улучшают существующие алгоритмы и включают их в будущие обновления программного обеспечения



В компьютерном программировании алгоритмы часто создаются как функции. Эти функции служат небольшими программами, на которые может ссылаться более крупная программа.

Например, для просмотра изображений приложение может включать в себя библиотеку функций, каждая из которых может использовать собственный алгоритм для отображения различных изображений форматов файлов.

Программа редактирования изображений может содержать алгоритмы, предназначенные для обработки данных изображения. Примеры алгоритмов обработки изображений включают кадрирование, изменение размера, повышение резкости, размытие, уменьшение эффекта красных глаз и улучшение цвета.

Во многих случаях существует несколько способов выполнить определенную операцию в программе. Поэтому программисты обычно стремятся создать максимально эффективные алгоритмы. Используя высокоэффективные алгоритмы, разработчики могут обеспечить максимально быструю работу своих программ и минимальное использование системных ресурсов. Конечно, не все алгоритмы создаются с первого раза идеально. Поэтому разработчики часто улучшают существующие алгоритмы и включают их в будущие обновления программного обеспечения. Когда вы видите новую версию программного обеспечения, которая была «оптимизирована» или имеет «более высокую производительность», это скорее всего означает, что новая версия включает более эффективные алгоритмы.