1. Примеры эффективных (полиномиальных) алгоритмов: быстрые алгоритмы поиска и сортировки; **Полиномиальные алгоритмы для задач на графах и сетях (поиск в глубину и ширину , о минимальном остове, о кратчайшем пути, о назначениях).**

В теории алгоритмов классом P (от англ. polynomial) называют множество алгоритмов, время работы которых не слишком сильно зависит от размера входных данных (не превосходит многочлена от размера данных). Алгоритмы, принадлежащие классу P, считаются быстрыми. Класс P включён в более широкие классы сложности алгоритмов.

Примерами алгоритмов класса P являются стандартные алгоритмы целочисленного сложения, умножения, деления, взятия остатка от деления, перемножения матриц, выяснение связности графов и некоторые другие.

Практическое значение

Поскольку часто приходится вычислять значения функций на входных данных большого объёма, нахождение полиномиальных алгоритмов для вычисления функций является очень важной задачей. Считается, что вычислять функции, не лежащие в классе P, заметно сложнее, чем лежащие. Большинство алгоритмов, лежащих в классе P, имеют сложность, не превосходящую многочлен небольшой степени от размера входных данных. Например, стандартный алгоритм перемножения матриц требует n3 умножений (хотя существуют и более быстрые алгоритмы, например, алгоритм Штрассена). Степень многочлена редко бывает большой. Один из таких случаев — найденный в 2002 г. индийскими математиками (Agrawal, Kayal и Saxena) алгоритм en:AKS primality test, выясняющий, является ли число простым, за O(log6n) операций, где n — длина записи числа в какой-нибудь системе счисления.

Поиск в глубину (англ. Depth-first search, DFS) — один из методов обхода графа. Алгоритм поиска описывается следующим образом: для каждой непройденной вершины необходимо найти все непройденные смежные вершины и повторить поиск для них. Используется в качестве подпрограммы в алгоритмах поиска одно- и двусвязных компонент, топологической сортировки.

**Алгоритм 1:**

Пусть задан [граф](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) G = (V, E), где V — множество вершин графа, E — множество ребер графа. Предположим, что в начальный момент времени все вершины графа окрашены в *белый* цвет. Выполним следующие действия:

1. Из множества всех *белых* вершин выберем любую вершину, обозначим её v_1.
2. Выполняем для неё процедуру DFS(v_1).
3. Повторяем шаги 1-3 до тех пор, пока множество *белых* вершин не пусто.

Процедура DFS (параметр — вершина u \in V)

1. Перекрашиваем вершину uв *черный* цвет.
2. Для всякой вершины w, [смежной](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2) с вершиной u и окрашенной в белый цвет, выполняем процедуру DFS(w).

**Поиск в глубину с метками времени. Классификация рёбер**

Для каждой из вершин установим два числа — «время» входа entry[u]и «время» выхода leave[u].

Модифицируем процедуру DFS так.

1. Увеличиваем «текущее время» на 1. entry[u] = t.
2. Перекрашиваем вершину uв чёрный цвет.
3. Для всякой вершины w, [смежной](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2) с вершиной u и окрашенной в белый цвет, выполняем процедуру DFS(w).
4. Увеличиваем «текущее время» на 1. leave[u] = t.

Считаем, что граф ориентированный. Очевидно, для любой вершины entry[u] < leave[u]. Просматриваемые на шаге 3 дуги *u*→*v* могут быть:

* entry[u] \leqslant t < entry[v] < leave[v] < leave[u]; в момент выполнения шага 3 entry[v]ещё не установлено. В таком случае мы для вершины *v* исполняем DFS, а дуга называется **дугой дерева поиска**.
* entry[u] < entry[v] < leave[v] \leqslant t < leave[u]; в момент выполнения шага 3 leave[v]уже установлено. Такая дуга называется **прямой**.
* entry[v] < leave[v] < entry[u] \leqslant t < leave[u]. Такая дуга называется **перекрёстной**.
* entry[v] < entry[u] \leqslant t < leave[u] < leave[v]; в момент выполнения шага 3 установлен только entry[v], но не leave. Имеем дело с **обратной** дугой.

Рёбра неориентированного графа могут быть рёбрами дерева, обратными и перекрёстными, но не прямыми. Чтобы различать рёбра неориентированного графа, достаточно трёхцветных отметок:

1. Перекрашиваем вершину uв **серый** цвет.
2. Для всякой вершины w, [смежной](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2) с вершиной u и окрашенной в белый цвет, выполняем процедуру DFS(w).
3. Перекрашиваем вершину uв **чёрный** цвет.

Ребро, идущее в белую вершину,— ребро дерева. В серую — обратное. В чёрную — перекрёстное.

[Алгоритм Косарайю](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%BE%D1%81%D0%B0%D1%80%D0%B0%D0%B9%D1%8E) требует сортировки вершин в обратном порядке по времени выхода. Метка входа и типы рёбер нужны в алгоритмах поиска [точек сочленения](http://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%BE%D1%87%D0%BA%D0%B0_%D1%81%D0%BE%D1%87%D0%BB%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B0&action=edit&redlink=1) и [мостов](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D1%81%D1%82_%28%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2%29&action=edit&redlink=1).

JAVA

public static void dfs(Node node, Node goal) { // Node - вершина графа

if (node.equals(goal)) {

System.out.println(node));

} else {

for (int i = 0; i < node.getNode().size(); i++) { // Вызываем этот же метод для каждой смежной вершины

if (stack.add(node.getNode().get(i))) { // Проверяем, вызывали ли мы этот метод для вершины node.getNode().get(i)

dfs(node.getNode().get(i), goal);

}

}

}

}

**Поиск в ширину** (**BFS**, Breadth-first search) — метод обхода и разметки вершин [графа](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29).

Поиск в ширину выполняется в следующем порядке: началу обхода s приписывается метка 0, смежным с ней вершинам — метка 1. Затем поочередно рассматривается окружение всех вершин с метками 1, и каждой из входящих в эти окружения вершин приписываем метку 2 и т. д.

Если исходный [граф](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) [связный](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), то поиск в ширину пометит все его вершины. Дуги вида (i, i+1) порождают остовный бесконтурный орграф, содержащий в качестве своей части остовное ордерево, называемое поисковым деревом.

Легко увидеть, что с помощью поиска в ширину можно также занумеровать вершины, нумеруя вначале вершины с меткой 1, затем с меткой 2 и т. д.

## Реализация

Поиск в ширину реализуется с помощью структуры [очередь](http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29). Для этого занесем в очередь исходную вершину. Затем будем работать, пока очередь не опустеет, таким образом: выберем элемент из очереди и добавим все смежные ему элементы, которые еще не использованы.

# Минимальное остовное дерево

## Алгоритмы

Существует несколько алгоритмов для нахождения минимального остовного дерева. Некоторые наиболее известные из них перечислены ниже:

* [Алгоритм Прима](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9F%D1%80%D0%B8%D0%BC%D0%B0);
* [Алгоритм Краскала](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D1%80%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D0%B0) (или алгоритм Крускала);
* [Алгоритм Борувки](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D1%80%D1%83%D0%B2%D0%BA%D0%B8).

**Алгори́тм Бору́вки** — это [алгоритм](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) нахождения [минимального остовного дерева](http://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE) в графе.

Впервые был опубликован в 1926 году Отакаром Борувкой в качестве метода нахождения оптимальной электрической сети в [Моравии](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%80%D0%B0%D0%B2%D0%B8%D1%8F). Несколько раз был переоткрыт, например [Флореком](http://ru.wikipedia.org/w/index.php?title=%D0%A4%D0%BB%D0%BE%D1%80%D0%B5%D0%BA,_%D0%9A%D0%B0%D0%B7%D0%B8%D0%BC%D0%B8%D1%80&action=edit&redlink=1), [Перкалом](http://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B5%D1%80%D0%BA%D0%B0%D0%BB,_%D0%94%D0%B6%D1%83%D0%BB%D0%B8%D0%B0%D0%BD&action=edit&redlink=1) и [Соллином](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BE%D0%BB%D0%BB%D0%B8%D0%BD&action=edit&redlink=1). Последний, кроме того, был единственным западным ученым из этого списка, и поэтому алгоритм часто называют **алгоритмом Соллина**, особенно в литературе по [параллельным вычислениям](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F).

## Алгоритм

Работа алгоритма состоит из нескольких итераций, каждая из которых состоит в последовательном добавлении рёбер к [остовному лесу](http://ru.wikipedia.org/wiki/%D0%9E%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BB%D0%B5%D1%81) графа, до тех пор, пока лес не превратится в [дерево](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%28%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2%29), то есть, лес, состоящий из одной компоненты связности.

В псевдокоде, алгоритм можно описать так:

1. Изначально, пусть T — пустое множество ребер (представляющее собой остовный лес, в который каждая вершина входит в качестве отдельного дерева).
2. Пока Tне является деревом (что эквивалентно условию: пока число рёбер в Tменьше, чем V - 1, где V — число вершин в графе):
   * Для каждой компоненты связности (то есть, дерева в остовном лесе) в подграфе с рёбрами T, найдём самое дешёвое ребро, связывающее эту компоненту с некоторой *другой* компонентой связности. (Предполагается, что веса рёбер различны, или как-то дополнительно упорядочены так, чтобы всегда можно было найти единственное ребро с минимальным весом).
   * Добавим все найденные рёбра в множество T.
3. Полученное множество рёбер Tявляется минимальным остовным деревом входного графа.

### Сложность алгоритма

На каждой итерации число деревьев в остовном лесу уменьшается по крайней мере в два раза, поэтому всего алгоритм совершает не более O(\log V)итераций. Каждая итерация может быть реализована со сложностью O(E), поэтому общее время работы алгоритмы составляет O(E \log V)времени (здесь Vи E — число вершин и рёбер в графе, соответственно).

Однако для некоторых видов графов, в частности, [планарных](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), оно может быть уменьшено до O(E).[[1]](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D1%80%D1%83%D0%B2%D0%BA%D0%B8#cite_note-1) Существует также рандомизированный алгоритм построения [минимального остовного дерева](http://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE), основанный на алгоритме Борувки, работающий в среднем за линейное время.

**Алгоритм Прима**  — [алгоритм](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) построения [минимального остовного дерева](http://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE) взвешенного связного неориентированного графа. Алгоритм впервые был открыт в 1930 году чешским математиком [Войцехом Ярником](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%BE%D0%B9%D1%86%D0%B5%D1%85_%D0%AF%D1%80%D0%BD%D0%B8%D0%BA&action=edit&redlink=1), позже переоткрыт [Робертом Примом](http://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%B8%D0%BC,_%D0%A0%D0%BE%D0%B1%D0%B5%D1%80%D1%82&action=edit&redlink=1) в 1957 году, и, независимо от них, [Э. Дейкстрой](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B8%D0%B1%D0%B5) в 1959 году.

## Описание

Построение начинается с дерева, включающего в себя одну (произвольную) вершину. В течение работы алгоритма дерево разрастается, пока не охватит все вершины исходного графа. На каждом шаге алгоритма к текущему дереву присоединяется самое лёгкое из рёбер, соединяющих вершину из построенного дерева и вершину не из дерева.

### Вход

* Связный неориентированный граф G(V,E)

### Выход

* Множество T рёбер минимального остовного дерева

## Реализация

### Обозначения

*  d[i]  — расстояние от i-й вершины до построенного дерева
*  p[i]  — предок i-й вершины, то есть такая вершина u, что (i,u)легчайшее из всех рёбер соединяющее i с вершиной из построенного дерева.
* w(i,j) — вес ребра (i,j)
* Q — приоритетная очередь вершин графа, где ключ — d[i]
* T — множество ребер минимального остовного дерева

### Псевдокод

 T \gets   {}

Для каждой вершины  i \in V    
 d[i] \gets \infty 

p[i] \gets nil

d[1] \gets 0   
Q \gets V   
v \gets\ Extract.min(Q)    
Пока Q  не пуста

Для каждой вершины u смежной с v

Если u \in Q и w(v,u) < d[u]

d[u] \gets w(v,u)

p[u] \gets v

 v \gets Extract.Min(Q)

 T \gets T+(p[v],v)

## Оценка

Асимптотика алгоритма зависит от способа хранения графа и способа хранения вершин, не входящих в дерево. Если приоритетная очередь Qреализована как обычный массив d, то Extract.Min(Q)выполняется за O(n), а стоимость операции d[u] \gets w(v, u)составляет O(1). Если Qпредставляет собой [бинарную пирамиду](http://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BA%D1%83%D1%87%D0%B0), то стоимость Extract.Min(Q)снижается до O(\log n), а стоимость d[u] \gets w(v,u)возрастает до O(\log n). При использовании [фибоначчиевых пирамид](http://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8%D0%B5%D0%B2%D0%B0_%D0%BA%D1%83%D1%87%D0%B0) операция Extract.Min(Q)выполняется за O(\log n), а d[u] \gets w(v, u)за O(1).

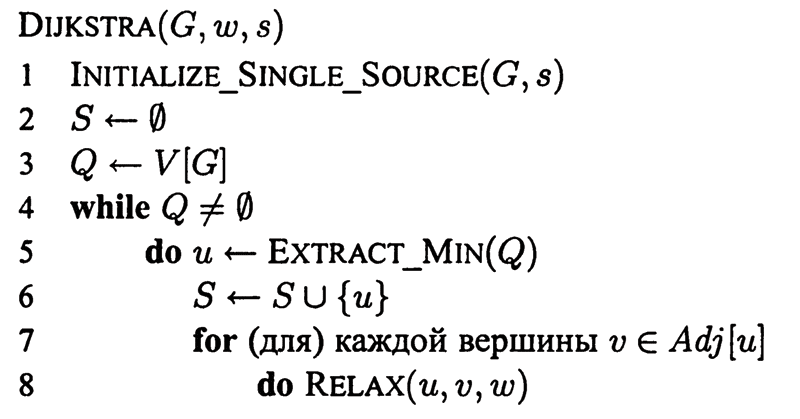
|  |  |
| --- | --- |
| **Способ представления графа и приоритетной очереди** | **Асимптотика** |
| Массив d, списки смежности (матрица смежности) | O(V^2) |
| Бинарная пирамида, списки смежности | O((V + E) \log V) = O(E \log V) |
| Фибоначчиева пирамида, списки смежности | O(E + V \log V) |

О кратчайшем пути:

## Алгоритм Дейкстры

**Алгоритм Дейкстры** — алгоритм на графах, изобретённый нидерландским ученым Э. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса.

### Формальное описание

  
В строке 1 производится обычная инициализация величин d и pi, а в строке 2 инициализируется пустое множество вершин S. В этом алгоритме поддерживается инвариант, согласно которому в начале каждой итерации цикла while в строках 4-8 выполняется равенство Q = V — S. В строке 3 неубывающая очередь с приоритетами Q инициализируется таким образом, чтобы она содержала все вершины множества V; поскольку в этот момент S = 0, после выполнения строки 3 сформулированный выше инвариант выполняется. При каждой итерации цикла while в строках 4-8 вершина и извлекается из множества Q = V — S и добавляется в множество 5, в результате чего инвариант продолжает соблюдаться. (Во время первой итерации этого цикла u = s.) Таким образом, вершина и имеет минимальную оценку кратчайшего пути среди всех вершин множества V — 5. Затем строках 7-8 ослабляются все ребра (u, v), исходящие из вершины и. Если текущий кратчайший путь к вершине v может быть улучшен в результате прохождения через вершину и, выполняется ослабление и соответствующее обновление оценки величины d [v] и предшественника тг [v]. Обратите внимание, что после выполнения строки 3 вершины никогда не добавляются в множество Q и что каждая вершина извлекается из этого множества и добавляется в множество 5 ровно по одному разу, поэтому количество итераций цикла while в строках 4-8 равно | V|. Поскольку в алгоритме Дейкстры из множества V — S для помещения в множество 5 всегда выбирается самая "легкая" или "близкая" вершина, говорят, что этот алгоритм придерживается жадной стратегии.

### Оценка сложности     [[вверх]](http://urban-sanjoo.narod.ru/dejkstra.html#tableTop)

Время выполнения алгоритма Дейкстры зависит от реализации неубывающей очереди с приоритетами. Сначала рассмотрим случай, когда неубывающая очередь с приоритетами поддерживается за счет того, что все вершины пронумерованы от 1 до |V|. Атрибут d[v] просто помещается в элемент массива с индексом v. Каждая операция Insert и Decrease\_Key (неявно присутствует в процедуре Relax) занимает время 0(1), а каждая операция Extract\_Min — время О (V) (поскольку в ней производится поиск по всему массиву); в результате полное **время работы алгоритма равно О (V2 + Е)** = О(V2).

Если граф достаточно разреженный, в частности, если количество вершин и ребер в нем связаны соотношением Е = o(V2/lgV), с практической точки зрения целесообразно реализовать неубывающую очередь с приоритетами в виде бинарной неубывающей пирамиды. (важная деталь реализации заключается в том, что вершины и соответствующие им элементы пирамиды должны поддерживать идентификаторы друг друга.) Далее, каждая операция Extract\_Min занимает время О (lg V), Как и раньше, таких операций |V|. Время, необходимое для построения неубывающей пирамиды, равно O(V). Каждая операция Decrease\_Key занимает время O(lgV), и всего выполняется не более |Е| таких операций. Поэтому полное время выполнения алгоритма равно О ((V + Е) lg V), что равно О (Е\*lg V), если все вершины достижимы из истока. Это время работы оказывается лучшим по сравнению со временем работы прямой реализации О (V2), если Е = о (V2/lg V).

**Задача о назначениях** – одна из фундаментальных задач [комбинаторной оптимизации](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) в области [математической оптимизации](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) или [исследовании операций](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B9). Задача состоит в поиске минимальной суммы дуг во [взвешенном](http://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%BE%D1%81%D1%81%D0%B0%D1%80%D0%B8%D0%B9_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2) [двудольном графе](http://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D0%B4%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84).

В наиболее общей форме задача формулируется следующим образом:

Имеется некоторое число *работ* и некоторое число *исполнителей*. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы так, чтобы выполнить работы с минимальными затратами.

Если число работ и исполнителей совпадает, то задача называется *линейной задачей о назначениях*. Обычно, если говорят о *задаче о назначениях* без дополнительных условий, имеют ввиду *линейную задачу о назначениях*.

## Алгоритмы и обобщения

[Венгерский алгоритм](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B3%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) – один из многих алгоритмов, разработанный для решения линейной задачи о назначениях за полиномиальное время от числа работ.

Задача о назначениях является частным случаем [транспортной задачи](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0), которая является частным случаем задачи нахождения [потока минимальной стоимости](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%BC%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D0%BE%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8), а она, в свою очередь, является частным случаем задачи [линейного программирования](http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Любую из этих задач можно решить [cимплекс-методом](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4), но каждая специализация имеет свой более эффективный алгоритм, опирающийся на особенности структуры задачи.

Если целевая функция выражается через квадраты, говорят о [квадратичной задаче о назначениях](http://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%D1%85&action=edit&redlink=1).

## Объяснение на примере

Предположим, есть три работника: Иван, Пётр и Андрей. Нужно распределить между ними выполнение трёх видов работ (которые мы назовём A, B, C), каждый работник должен выполнять только одну разновидность работ. Как это сделать так, чтобы потратить наименьшую сумму денег на оплату труда рабочих? Сначала необходимо построить [матрицу](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) стоимостей работ.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **A** | **B** | **C** |
| Иван | 10.000 руб. | 20.000 руб. | 30.000 руб. |
| Пётр | 30.000 руб. | 30.000 руб. | 30.000 руб. |
| Андрей | 30.000 руб. | 30.000 руб. | 20.000 руб. |

Венгерский алгоритм, применённый к приведённой выше таблице даст нам требуемое распределение: Иван выполняет работу A, Пётр — работу B, Андрей — работу С.

## Постановка задачи

Дана неотрицательная [матрица](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) размера *n*×*n*, где элемент в *i*-й строке и *j*-ом столбце соответствует стоимости выполнения *j*-ого вида работ *i*-м работником. Нужно найти такое соответствие работ работникам, чтобы расходы на оплату труда были наименьшими. Если цель состоит в нахождении назначения с наибольшей стоимостью, то решение сводится к решению только что сформулированной задачи путём замены каждой стоимости *C* на разность между максимальной стоимостью и *C*.[[2]](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B3%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC#cite_note-2)

## Основные идеи

Алгоритм основан на двух идеях:

* если из всех элементов некой строки или столбца вычесть одно и то же число y, общая стоимость уменьшится на y, а оптимальное решение не изменится;
* если есть решение нулевой стоимости, оно оптимально.

Алгоритм ищет значения, которые надо вычесть из всех элементов каждой строки и каждого столбца (разные для разных строк и столбцов), такие что все элементы матрицы останутся неотрицательными, но появится нулевое решение.

1. **Булевы функции, канонические формы задания булевых функций.**

**Бу́лева фу́нкция** (или **логи́ческая функция**, или **функция а́лгебры ло́гики**)[[1]](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F#cite_note-.D0.98.D0.B3.D0.BE.D1.88.D0.B8.D0.BD.E2.80.942008.E2.80.94.D0.93.D0.BB.D0.B0.D0.B2.D0.B0_2._.D0.91.D1.83.D0.BB.D0.B5.D0.B2.D1.8B_.D1.84.D1.83.D0.BD.D0.BA.D1.86.D0.B8.D0.B8.E2.80.94-1) от *n* аргументов — в [дискретной математике](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) — отображение *Bn* → *B*, где *B* = {0,1} — *булево множество*. Элементы булева множества {1, 0} обычно интерпретируют как логические значения «истинно» и «ложно», хотя в общем случае они рассматриваются как формальные символы, не несущие определённого смысла. Неотрицательное целое число *n* называют [арностью](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C) или местностью функции, в случае *n* = 0 булева функция превращается в *булеву константу*. Элементы [декартова произведения](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%B0%D1%80%D1%82%D0%BE%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5) (*n*-я прямая степень) *Bn* называют *булевыми векторами*. Множество всех булевых функций от любого числа аргументов часто обозначается *P*2, а от *n* аргументов — *P*2(*n*). Переменные, принимающие значения из булева множества называются *булевыми переменными*[[2]](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F#cite_note-.D0.98.D0.B3.D0.BE.D1.88.D0.B8.D0.BD.E2.80.942008.E2.80.94.E2.80.9494-2). Булевы функции названы по фамилии математика [Джорджа Буля](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D1%8C,_%D0%94%D0%B6%D0%BE%D1%80%D0%B4%D0%B6).

При работе с булевыми функциями происходит полное абстрагирование от содержательного смысла, который имелся в виду в алгебре высказываний[[2]](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F#cite_note-.D0.98.D0.B3.D0.BE.D1.88.D0.B8.D0.BD.E2.80.942008.E2.80.94.E2.80.9494-2). Тем не менее, между булевыми функциями и формулами [алгебры высказываний](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0_%D0%B2%D1%8B%D1%81%D0%BA%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9) можно установить взаимно-однозначное соответствие, если[[3]](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F#cite_note-.D0.98.D0.B3.D0.BE.D1.88.D0.B8.D0.BD.E2.80.942008.E2.80.94.E2.80.94104-105-3):

* установить взаимно-однозначное соответствие между булевыми переменными и пропозициональными переменными,
* установить связь между булевыми функциями и логическими связками,
* оставить расстановку скобок без изменений.

## Представление булевых функций

Основная статья: [**Представление булевых функций**](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B1%D1%83%D0%BB%D0%B5%D0%B2%D1%8B%D1%85_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B9)

Теорема Поста открывает путь к представлению булевых функций синтаксическим способом, который в ряде случаев оказывается намного удобнее, чем таблицы истинности. Отправной точкой здесь служит нахождение некоторой полной системы функций \Sigma = \{f_1,\ldots,f_n\}. Тогда каждая булева функция может быть представлена некоторым [термом](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%BC_%28%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0%29) в [сигнатуре](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D1%82%D1%83%D1%80%D0%B0_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0%29) \Sigma, который в данном случае называют также [формулой](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B0). Относительно выбраной системы функций полезно знать ответы на следующие вопросы:

* Как построить по данной функции представляющую её формулу?
* Как проверить, что две разные формулы эквивалентны, то есть задают одну и ту же функцию?
  + В частности: существует ли способ приведения произвольной формулы к эквивалентной ей *канонической* форме такой, что две формулы эквивалентны тогда и только тогда, когда их канонические формы совпадают?
* Как по данной функции построить представляющую её формулу с теми или иными заданными свойствами (например, наименьшего размера), и возможно ли это?

Положительные ответы на эти и другие вопросы существенно увеличивают прикладное значение выбранной системы функций.

### Дизъюнктивная нормальная форма (ДНФ)

Основная статья: [**Дизъюнктивная нормальная форма**](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B7%D1%8A%D1%8E%D0%BD%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)

*Простой конъюнкцией* или *конъюнктом* называется конъюнкция некоторого конечного набора переменных или их отрицаний, причём каждая переменная встречается не более одного раза. *Дизъюнктивной нормальной формой* или *ДНФ* называется дизъюнкция простых конъюнкций. Элементарная конъюнкция

* **правильная**, если каждая переменная входит в неё не более одного раза (включая отрицание);
* **полная**, если каждая переменная (или её отрицание) входит в неё ровно 1 раз;
* **монотонная**, если она не содержит отрицаний переменных.

Например a \overline{b} c\lor b c\lor\overline{a}  — является ДНФ.

*Совершенной дизъюнктивной нормальной формой* или *СДНФ* относительно некоторого заданного конечного набора переменных называется такая ДНФ, у которой в каждую конъюнкцию входят все переменные данного набора, причём в одном и том же порядке. Например: a \overline{b} c\lor a b c\lor\overline{a} b\overline{c}.

Легко убедиться, что каждой булевой функции соответствует некоторая ДНФ, а функции, отличной от тождественного нуля — даже СДНФ. Для этого достаточно в таблице истинности этой функции найти все булевы векторы, на которых её значение равно 1, и для каждого такого вектора b=(b_1,b_2,\ldots,b_n)построить конъюнкцию x_1^{b_1} x_2^{b_2}\ldots x_n^{b_n}, где x_i^1 = x_ix_i^0 = \overline{x_i}. Дизъюнкция этих конъюнкций является СДНФ исходной функции, поскольку на всех булевых векторах её значения совпадают со значениями исходной функции. Например, для импликации x\to yрезультатом является \overline{x} y \lor \overline{x}\, \overline{y}\lor x y, что можно упростить до \overline{x}\lor y.

### Конъюнктивная нормальная форма (КНФ)

Основная статья: [**Конъюнктивная нормальная форма**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%8A%D1%8E%D0%BD%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)

*Конъюнктивная нормальная форма1* (КНФ) определяется двойственно к ДНФ. *Простой дизъюнкцией* или *дизъюнктом* называется дизъюнкция одной или нескольких переменных или их отрицаний, причём каждая переменная входит в неё не более одного раза. КНФ — это конъюнкция простых дизъюнкций.

*Совершенной конъюнктивной нормальной формой* (СКНФ), относительно некоторого заданного конечного набора переменных, называется такая КНФ, у которой в каждую дизъюнкцию входят все переменные данного набора, причём в одном и том же порядке. Поскольку (С)КНФ и (С)ДНФ взаимодвойственны, свойства (С)КНФ повторяют все свойства (С)ДНФ, грубо говоря, «с точностью до наоборот».

КНФ может быть преобразована к эквивалентной ей ДНФ путём раскрытия скобок по правилу:

a (b\lor c)\to a b\lor a c

которое выражает [дистрибутивность](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) конъюнкции относительно дизъюнкции. После этого необходимо в каждой конъюнкции удалить повторяющиеся переменные или их отрицания, а также выбросить из дизъюнкции все конъюнкции, в которых встречается переменная вместе со своим отрицанием. При этом результатом не обязательно будет СДНФ, даже если исходная КНФ была СКНФ. Точно также можно всегда перейти от ДНФ к КНФ. Для этого следует использовать правило

a\lor b c\to (a \lor b)(a \lor c)

выражающее дистрибутивность дизъюнкции относительно конъюнкции. Результат нужно преобразовать описанным выше способом, заменив слово «конъюнкция» на «дизъюнкция» и наоборот.

### Алгебраическая нормальная форма (АНФ или полином Жегалкина)

Основная статья: [**Полином Жегалкина**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC_%D0%96%D0%B5%D0%B3%D0%B0%D0%BB%D0%BA%D0%B8%D0%BD%D0%B0)

[Алгебраическая нормальная форма](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) (общепринятое название в зарубежной литературе) или [полином Жегалкина](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC_%D0%96%D0%B5%D0%B3%D0%B0%D0%BB%D0%BA%D0%B8%D0%BD%D0%B0) (название, используемое в отечественной литературе) — это форма представления логической функции в виде [полинома](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC) с коэффициентами вида 0 и 1, в котором в качестве произведения используется операция [конъюнкции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%8A%D1%8E%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) («И», AND), а в качестве сложения — [сложение по модулю 2](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2) (исключающее «ИЛИ», XOR). Для получения полинома Жегалкина следует выполнить следующие действия:

1. Получить СДНФ функции
2. Все ИЛИ заменить на Исключающее ИЛИ
3. Во всех термах заменить элементы с отрицанием на конструкцию: («элемент» «исключающее ИЛИ» 1)
4. Раскрыть скобки по правилам алгебры Жегалкина и привести попарно одинаковые термы

## Классификация булевых функций

* По количеству *n* входных операндов, от которых зависит значение на выходе функции, различают нульарные (*n* = 0), унарные (*n* = 1), бинарные (*n* = 2), тернарные (*n* = 3) булевы функции и функции от большего числа операндов.
* По количеству единиц и нулей в таблице истинности отличают узкий класс [сбалансированных булевых функций](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) (также называемых уравновешенными или равновероятноятными, поскольку при равновероятных случайных значениях на входе или при переборе всех комбинаций по таблице истинности вероятность получения на выходе значения **1** равна 1/2) от более широкого класса несбалансированных булевых функций (так же называемых неуравновешенными, поскольку вероятность получения на выходе значения **1** отлична от 1/2). Сбалансированные булевы функции в основном используются в [криптографии](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F).
* По зависимости значения функции от перестановки её входных битов различают [симметричные булевы функции](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%B1%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) (значение которых зависит только от количества единиц на входе) и несимметричные булевы функции (значение которых так же зависит от перестановки её входных бит).
* По значению функции на противоположных друг другу наборах значений аргументов отличают [самодвойственные функции](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%B4%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) (значение которых инвертируется при инвертировании значения всех входов) от остальных булевых функций, не обладающих таким свойством. Нижняя часть [таблицы истинности](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_%D0%B8%D1%81%D1%82%D0%B8%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8) для самодвойственных функций является зеркальным отражением инвертированной верхней части (если расположить входные комбинации в таблице истинности в естественном порядке).
* По алгебраической степени нелинейности отличают [линейные булевы функции](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) ([АНФ](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) которых сводится к линейной сумме по модулю 2 входных значений) и [нелинейные булевы функции](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) ([АНФ](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) которых содержит хотя бы одну нелинейную операцию конъюнкции входных значений). Примерами линейных функций являются: [сложение по модулю 2](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2) (исключающее «ИЛИ», XOR), [эквивалентность](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D0%B2%D0%B8%D0%B2%D0%B0%D0%BB%D0%B5%D0%BD%D1%86%D0%B8%D1%8F), а также все булевы функции, [АНФ](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) которых содержит лишь линейные операции сложения по модулю 2 без конъюнкций. Примерами нелинейных функций являются: [конъюнкция](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%8A%D1%8E%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) («И», AND), [штрих Шеффера](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D1%80%D0%B8%D1%85_%D0%A8%D0%B5%D1%84%D1%84%D0%B5%D1%80%D0%B0) («НЕ-И», NAND), [стрелка Пирса](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%BA%D0%B0_%D0%9F%D0%B8%D1%80%D1%81%D0%B0) («НЕ-ИЛИ», NOR), а также все булевы функции, [АНФ](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) которых содержит хотя бы одну нелинейную операцию конъюнкции.

1. **Формальные языки и способы их описания.**

В [математической логике](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) и [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) **формальный язык** — это [множество](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) конечных [слов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%BE_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) (строк, [цепочек](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6)) над конечным [алфавитом](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29#.D0.92_.D0.BC.D0.B0.D1.82.D0.B5.D0.BC.D0.B0.D1.82.D0.B8.D0.BA.D0.B5). Понятие языка чаще всего используется в [теории автоматов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%B2), [теории вычислимости](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) и [теории алгоритмов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2). Научная теория, которая имеет дело с этим объектом, называется [теорией формальных языков](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D1%85_%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2&action=edit&redlink=1).

В [теории моделей](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B5%D0%B9) **язык** соответствует не языку в информатике, а скорее алфавиту. Язык состоит из множеств символов, [функций](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) и [отношений](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) вместе с их [арностью](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C), а также множество [переменных](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0). Каждое из этих множеств может быть бесконечным. Из языка вместе с [универсальными логическими символами](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D1%8B%D1%81%D0%BA%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9) составляются логические высказывания.

## Определение

Формальный язык может быть определён по-разному, например:

* Простым перечислением слов, входящих в данный язык. Этот способ, в основном, применим для определения конечных языков и языков простой структуры.
* Словами, порождёнными некоторой [формальной грамматикой](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) (см. [иерархия Хомского](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)).
* Словами, порождёнными [регулярным выражением](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F).
* Словами, распознаваемыми некоторым [конечным автоматом](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82).
* Словами, порождёнными [БНФ-конструкцией](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0_%D0%91%D1%8D%D0%BA%D1%83%D1%81%D0%B0_%E2%80%94_%D0%9D%D0%B0%D1%83%D1%80%D0%B0).

Если алфавит задан как {*a*, *b*}, а язык *L* включает в себя все слова над ним, то слово *ababba* принадлежит *L*. **Пустое слово** (то есть строка нулевой длины) допускается и часто обозначается как *e*, ε или Λ.

Некоторые примеры формальных языков:

* множество всех слов над {*a*, *b*}
* множество \{ a^n\}, где *n* — [неотрицательное число](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BE%D1%82%D1%80%D0%B8%D1%86%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE), а a^{n}означает, что *a* повторяется *n* раз
* множество [синтаксически](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81) корректных [программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) в данном [языке программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)

## Операции

Некоторые операции могут быть использованы для того, чтобы порождать новые языки из данных. Предположим, что L_{1}и L_{2}являются языками, определёнными над некоторым общим алфавитом.

* *Конкатенация (сцепление)* L_{1}L_{2}содержит все слова, удовлетворяющие форме vw, где v— это слово из L_{1}, а w— слово из L_{2}.
* *Пересечение* L_1 \cap L_2содержит все слова, содержащиеся и в L_1, и в L_{2}.
* *Объединение* L_1 \cup L_2содержит все слова, содержащиеся или в L_{1}, или в L_{2}.
* *Дополнение* языка L_{1}содержит все слова алфавита, которые не содержатся в L_{1}.
* *Правое отношение* L_{1}/L_{2}содержит все слова v, для которых существует слово wв L_{2}такое, что vwнаходилось в L_{1}.
* [*Замыкание Клини*](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_%D0%9A%D0%BB%D0%B8%D0%BD%D0%B8) L_{1}^{*}содержит все слова, которые могут быть записаны в форме w_{1}w_{2}...w_{n}, где w_{i}содержится в L_{1}и n \ge 0. Следует помнить, что это включает и пустое слово \epsilon, так как n = 0допустимо по условию.
* *Обращение* L_{1}^{R}содержит обращённые слова из L_{1}.
* *Смешение* L_{1}и L_{2}содержит все слова, которые могут быть записаны в форме v_{1}w_{1}v_{2}w_{2}...v_{n}w_{n}, где n \ge 1и v_{1},...,v_{n}являются такими словами, что связь v_{1}...v_{n}находится в L_{1}, а w_{1},...,w_{n}являются такими словами, что w_{1}...w_{n}находятся в L_{2}.

**Иерархия Хомского** — классификация [формальных языков](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) и [формальных грамматик](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), согласно которой они делятся на 4 типа по их условной сложности. Предложена профессором [Массачусетского технологического института](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B0%D1%87%D1%83%D1%81%D0%B5%D1%82%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82), лингвистом [Ноамом Хомским](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%B8%D0%B9,_%D0%90%D0%B2%D1%80%D0%B0%D0%BC_%D0%9D%D0%BE%D0%B0%D0%BC).

## Классификация грамматик

Согласно Хомскому, [формальные грамматики](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) делятся на четыре типа. Для отнесения грамматики к тому или иному типу необходимо соответствие *всех* её правил (продукций) некоторым схемам.

### Тип 0 — неограниченные

*Грамматика с фразовой структурой* G — это [алгебраическая структура](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0), упорядоченная четвёрка (VT, VN, P, S), где[[1]](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE#cite_note-.D0.9A.D1.83.D0.BA.2C_.D0.91.D0.B5.D0.B9.D0.B7.E2.80.941990.E2.80.94.E2.80.94258.2C264-1):

* V_T — *алфавит* (множество) терминальных символов — *терминалов*,
* V_N — *алфавит* (множество) нетерминальных символов — *нетерминалов*,
* V = V_T \cup V_N — *словарь* G, причём V_T \cap V_N = \empty
* P — *конечное множество продукций* (правил) грамматики, P \subseteq V^+\times V^*
* S — *начальный символ* (*источник*).

Здесь V^* — множество всех строк над алфавитом V, а V^+ — множество непустых строк над алфавитом V.

К типу 0 по классификации Хомского относятся [неограниченные грамматики](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8) — [грамматики с фразовой структурой](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0_%D1%81_%D1%84%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BE%D0%B9), то есть все без исключения формальные грамматики. Правила можно записать в виде:

\alpha\rarr\beta,

где \alpha — любая непустая цепочка, содержащая хотя бы один нетерминальный[[*источник не указан 287 дней*]](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8) символ, а \beta — любая цепочка символов из алфавита.

Практического применения в силу своей сложности такие грамматики не имеют.

### Тип 1 — контекстно-зависимые

К этому типу относятся [контекстно-зависимые (КЗ) грамматики](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) и неукорачивающие грамматики. Для грамматики G(VT,VN,P, S), V=VT∪VN все правила имеют вид[[2]](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE#cite_note-.D0.9A.D1.83.D0.BA.2C_.D0.91.D0.B5.D0.B9.D0.B7.E2.80.941990.E2.80.94.E2.80.94268-2):

* αAβ→αγβ, где α, β∈V\*, γ∈V+, A∈VN. Такие грамматики относят к контекстно-зависимым.
* α→β, где α, β∈V+, 1≤|α|≤|β|. Такие грамматики относят к неукорачивающим.

Эти классы грамматик эквивалентны. Могут использоваться при анализе текстов на естественных языках, однако при построении компиляторов практически не используются в силу своей сложности. Для контекстно-зависимых грамматик доказано утверждение: по некоторому алгоритму за конечное число шагов можно установить, принадлежит цепочка терминальных символов данному языку или нет.

### Тип 2 — контекстно-свободные

К этому типу относятся [контекстно-свободные (КС) грамматики](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0). Для грамматики G(VT,VN,P, S), V=VT∪VN все правила имеют вид:

* A→β, где β∈V+ (для неукорачивающих КС-грамматик, β∈V\* для укорачивающих), A∈VN. То есть грамматика допускает появление в левой части правила только [нетерминального символа](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB&action=edit&redlink=1).

КС-грамматики широко применяются для описания синтаксиса [компьютерных языков](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) (см. [синтаксический анализ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7)).

### Тип 3 — регулярные

К третьему типу относятся [регулярные грамматики](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8) (автоматные) — самые простые из формальных грамматик. Они являются контекстно-свободными, но с ограниченными возможностями.

Все регулярные грамматики могут быть разделены на два эквивалентных класса, которые для грамматики вида III будут иметь правила следующего вида:

* A→Bγ или A→γ, где γ∈VT\*, A, B∈VN (для леволинейных грамматик).
* A→γB; или A→γ, где γ∈VT\*, A, B∈VN (для праволинейных грамматик).

Регулярные грамматики применяются для описания простейших конструкций: [идентификаторов](https://ru.wikipedia.org/wiki/%D0%98%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80), [строк](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), [констант](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), а также [языков ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0), [командных процессоров](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) и др.

## Классификация языков

[Формальные языки](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) классифицируются в соответствии с типами грамматик, которыми они задаются. Однако, один и тот же язык может быть задан разными грамматиками, относящимися к разным типам. В таком случае, считается, что язык относится к наиболее простому из них. Так, язык, описанный грамматикой с фразовой структурой, контекстно-зависимой и контекстно-свободной грамматиками, будет контекстно-свободным.

Так же, как и для грамматик, сложность языка определяется его типом. Наиболее сложные — языки с фразовой структурой (сюда можно отнести естественные языки), далее — КЗ-языки, КС-языки и самые простые — регулярные языки.

1. **Методы сжатия информации.**

**Сжатие данных** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *data compression*) — алгоритмическое преобразование [данных](http://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), производимое с целью уменьшения занимаемого ими объёма. Применяется для более рационального использования устройств [хранения](http://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) и [передачи](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) данных. [Синонимы](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC) — *упаковка данных*, *компрессия*, *сжимающее кодирование*, [*кодирование*](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) *источника*. Обратная процедура называется восстановлением данных (распаковкой, декомпрессией).

Сжатие основано на устранении [избыточности](http://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), содержащейся в исходных данных. Простейшим примером избыточности является повторение в [тексте](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82) фрагментов (например, слов естественного или машинного языка). Подобная избыточность обычно устраняется заменой повторяющейся последовательности ссылкой на уже закодированный фрагмент с указанием его длины. Другой вид избыточности связан с тем, что некоторые значения в сжимаемых данных встречаются чаще других. Сокращение объёма данных достигается за счёт замены часто встречающихся данных короткими кодовыми словами, а редких — длинными ([энтропийное кодирование](http://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Энтропийное кодирование)). Сжатие данных, не обладающих свойством избыточности (например, [случайный сигнал](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB) или [белый шум](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BB%D1%8B%D0%B9_%D1%88%D1%83%D0%BC), зашифрованные сообщения), принципиально невозможно без потерь.

## Принципы сжатия данных

В основе любого способа сжатия лежит модель источника данных, или, точнее, [модель избыточности](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B8%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1). Иными словами, для сжатия данных используются некоторые априорные сведения о том, какого рода данные сжимаются. Не обладая такими сведениями об источнике, невозможно сделать никаких предположений о преобразовании, которое позволило бы уменьшить объём сообщения. Модель избыточности может быть статической, неизменной для всего сжимаемого сообщения, либо строиться или параметризоваться на этапе сжатия (и восстановления). Методы, позволяющие на основе входных данных изменять модель избыточности информации, называются адаптивными. Неадаптивными являются обычно узкоспециализированные алгоритмы, применяемые для работы с данными, обладающими хорошо определёнными и неизменными характеристиками. Подавляющая часть достаточно универсальных алгоритмов являются в той или иной мере адаптивными.

Все методы сжатия данных делятся на два основных класса:

* [Сжатие без потерь](http://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C)
* [Сжатие с потерями](http://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D1%81_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8F%D0%BC%D0%B8)

При использовании сжатия без потерь возможно полное восстановление исходных данных, сжатие с потерями позволяет восстановить данные с искажениями, обычно несущественными с точки зрения дальнейшего использования восстановленных данных. Сжатие без потерь обычно используется для передачи и хранения текстовых данных, компьютерных программ, реже — для сокращения объёма [аудио](http://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%B4%D0%B8%D0%BE)- и [видеоданных](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE), [цифровых фотографий](http://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%84%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и т. п., в случаях, когда искажения недопустимы или нежелательны. Сжатие с потерями, обладающее значительно большей, чем сжатие без потерь, эффективностью, обычно применяется для сокращения объёма аудио- и видеоданных и цифровых фотографий в тех случаях, когда такое сокращение является приоритетным, а полное соответствие исходных и восстановленных данных не требуется.

## Характеристики алгоритмов сжатия и их применимость

### Коэффициент сжатия

Коэффициент сжатия — основная характеристика алгоритма сжатия. Она определяется как отношение объёма исходных несжатых данных к объёму сжатых, то есть: k=\frac{S_o}{S_c}, где *k* — коэффициент сжатия, *S*o — объём исходных данных, а *S*c — объём сжатых. Таким образом, чем выше коэффициент сжатия, тем алгоритм эффективнее. Следует отметить:

* если *k* = 1, то алгоритм не производит сжатия, то есть выходное сообщение оказывается по объёму равным входному;
* если *k* < 1, то алгоритм порождает сообщение большего размера, нежели несжатое, то есть, совершает «вредную» работу.

Ситуация с *k* < 1 вполне возможна при сжатии. Принципиально невозможно получить алгоритм сжатия без потерь, который при любых данных образовывал бы на выходе данные меньшей или равной длины. Обоснование этого факта заключается в том, что поскольку число различных сообщений длиной *n* бит составляет ровно 2*n*, число различных сообщений с длиной меньшей или равной *n* (при наличии хотя бы одного сообщения меньшей длины) будет меньше 2*n*. Это значит, что невозможно однозначно сопоставить все исходные сообщения сжатым: либо некоторые исходные сообщения не будут иметь сжатого представления, либо нескольким исходным сообщениям будет соответствовать одно и то же сжатое, а значит их нельзя отличить. Однако даже когда алгоритм сжатия увеличивает размер исходных данных, легко добиться того, чтобы их объём гарантировано не мог увеличиться более, чем на 1 бит. Тогда даже в самом худшем случае будет иметь место неравенство:  
k\geqslant\frac{S_o}{S_o+1}  
Делается это следующим образом: если объём сжатых данных меньше объёма исходных, возвращаем сжатые данные, добавив к ним «1», иначе возвращаем исходные данные, добавив к ним «0»). Пример того, как это реализуется на псевдо-[C++](http://ru.wikipedia.org/wiki/C%2B%2B), показан ниже:

bin\_data\_t \_\_compess(bin\_data\_t input) // bin\_data\_t - тип данных, означающий произвольную последовательность бит переменной длины

{

bin\_data\_t output = arch(input); // функция bin\_data\_t arch(bin\_data\_t input) реализует некий алгоритм сжатия данных

if (output.size()<input.size()) // если объём сжатых данных меньше объёма исходных, функция bin\_data\_t::size() возвращает размер данных

{

output.add\_begin(1); // функция bin\_data\_t::add\_begin(bool \_\_bit\_\_) добавляет бит, равный \_\_bit\_\_ в начало последовательности

return output; // возвращаем сжатую последовательность с добавленной «1»

}

else // иначе (если объём сжатых данных больше или равен объёму исходных)

{

input.add\_begin(0); // добавляем «0» к исходной последовательности

return input; // возвращаем исходный файл с добавленным «0»

}

}

Коэффициент сжатия может быть как постоянным (некоторые алгоритмы сжатия звука, изображения и т. п., например [А-закон](http://ru.wikipedia.org/wiki/%D0%90-%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD), [μ-закон](http://ru.wikipedia.org/wiki/%D0%9C%D1%8E-%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD), [ADPCM](http://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D1%83%D0%BB%D1%8C%D1%81%D0%BD%D0%BE-%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F), [усечённое блочное кодирование](http://ru.wikipedia.org/w/index.php?title=%D0%A3%D1%81%D0%B5%D1%87%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B1%D0%BB%D0%BE%D1%87%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)), так и переменным. Во втором случае он может быть определён либо для каждого конкретного сообщения, либо оценён по некоторым критериям:

* средний (обычно по некоторому тестовому набору данных);
* максимальный (случай наилучшего сжатия);
* минимальный (случай наихудшего сжатия);

или каким-либо другим. Коэффициент сжатия с потерями при этом сильно зависит от допустимой погрешности сжатия или *качества*, которое обычно выступает как параметр алгоритма. В общем случае постоянный коэффициент сжатия способны обеспечить только методы сжатия данных с потерями.

### Допустимость потерь

Основным критерием различия между алгоритмами сжатия является описанное выше наличие или отсутствие потерь. В общем случае алгоритмы сжатия без потерь универсальны в том смысле, что их применение безусловно возможно для данных любого типа, в то время как возможность применения сжатия с потерями должна быть обоснована. Для некоторых типов данных искажения не допустимы в принципе. В их числе

* символические данные, изменение которых неминуемо приводит к изменению их семантики: программы и их исходные тексты, двоичные массивы и т. п.;
* жизненно важные данные, изменения в которых могут привести к критическим ошибкам: например, получаемые с медицинской измерительной аппаратуры или контрольных приборов летательных, космических аппаратов и т. п.;
* многократно подвергаемые сжатию и восстановлению промежуточные данные при многоэтапной обработке графических, звуковых и видеоданных.

### Системные требования алгоритмов

Различные алгоритмы могут требовать различного количества ресурсов вычислительной системы, на которых они реализованы:

* оперативной памяти (под промежуточные данные);
* постоянной памяти (под код программы и константы);
* процессорного времени.

В целом, эти требования зависят от сложности и «интеллектуальности» алгоритма. Общая тенденция такова: чем эффективнее и универсальнее алгоритм, тем большие требования к вычислительным ресурсам он предъявляет. Тем не менее, в специфических случаях простые и компактные алгоритмы могут работать не хуже сложных и универсальных. Системные требования определяют их потребительские качества: чем менее требователен алгоритм, тем на более простой, а следовательно, компактной, надёжной и дешёвой системе он может быть реализован.

Так как алгоритмы сжатия и восстановления работают в паре, имеет значение соотношение системных требований к ним. Нередко можно усложнив один алгоритм значительно упростить другой. Таким образом, возможны три варианта:

Алгоритм сжатия требует больших вычислительных ресурсов, нежели алгоритм восстановления.

Это наиболее распространённое соотношение, характерное для случаев, когда однократно сжатые данные будут использоваться многократно. В качестве примера можно привести цифровые аудио- и видеопроигрыватели.

Алгоритмы сжатия и восстановления требуют приблизительно равных вычислительных ресурсов.

Наиболее приемлемый вариант для линий связи, когда сжатие и восстановление происходит однократно на двух её концах (например, в цифровой телефонии).

Алгоритм сжатия существенно менее требователен, чем алгоритм восстановления.

Такая ситуация характерна для случаев, когда процедура сжатия реализуется простым, часто портативным устройством, для которого объём доступных ресурсов весьма критичен, например, космический аппарат или большая распределённая сеть датчиков. Это могут быть также данные, распаковка которых требуется в очень малом проценте случаев, например запись камер видеонаблюдения.

## Алгоритмы сжатия данных неизвестного формата

Имеется два основных подхода к сжатию данных неизвестного формата:

* На каждом шаге [алгоритма](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2) сжатия очередной сжимаемый символ либо помещается в выходной буфер сжимающего кодера как есть (со специальным флагом, помечающим, что он не был сжат), либо группа из нескольких сжимаемых символов заменяется ссылкой на совпадающую с ней группу из уже закодированных символов. Поскольку восстановление сжатых таким образом данных выполняется очень быстро, такой подход часто используется для создания самораспаковывающихся программ.
* Для каждой сжимаемой последовательности символов однократно либо в каждый момент времени собирается статистика её встречаемости в кодируемых данных. На основе этой статистики вычисляется вероятность значения очередного кодируемого символа (либо последовательности символов). После этого применяется та или иная разновидность [энтропийного кодирования](http://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), например, [арифметическое кодирование](http://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) или [кодирование Хаффмана](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0), для представления часто встречающихся последовательностей короткими кодовыми словами, а редко встречающихся — более длинными.

### [Алгоритмы сжатия с потерями](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F_%D1%81_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8F%D0%BC%D0%B8)

* [Линейное предсказывающее кодирование](http://ru.wikipedia.org/w/index.php?title=%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D0%BA%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Linear_predictive_coding)) — сжатие с потерями, представляющее спектральную огибающую цифрового сигнала речи в сжатом виде
* [А-закон](http://ru.wikipedia.org/wiki/%D0%90-%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD) — стандартный алгоритм компандирования. Применяется в РФ.
* [Мю-закон](http://ru.wikipedia.org/wiki/%D0%9C%D1%8E-%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD) — стандартный алгоритм компандирования
* [Фрактальное сжатие](http://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D0%B5) — метод, использующий фракталы для сжатия изображений
* [Трансформирующее кодирование](http://ru.wikipedia.org/w/index.php?title=%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D1%84%D0%BE%D1%80%D0%BC%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B5%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Transform_coding)) — тип сжатия данных для «естественных» данных, таких как аудиосигналы или фотографические изображения
* [Векторное квантование](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%BA%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — техника, часто используемая в сжатии данных с потерями
* [Вейвлетное сжатие](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B9%D0%B2%D0%BB%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D0%B5) — тип компрессии данных хорошо подходящий для сжатия изображений (иногда также используется для сжатия видео и аудио)

### [Алгоритмы сжатия без потерь](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C)

* [Преобразование Барроуза — Уилера](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%91%D0%B0%D1%80%D1%80%D0%BE%D1%83%D0%B7%D0%B0_%E2%80%94_%D0%A3%D0%B8%D0%BB%D0%B5%D1%80%D0%B0) (также известен как [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *BWT*) — предварительная обработка данных для улучшения сжатия без потерь
* [Преобразование Шиндлера](http://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A8%D0%B8%D0%BD%D0%B4%D0%BB%D0%B5%D1%80%D0%B0&action=edit&redlink=1) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)  *ST*) — модификация преобразования Барроуза — Уилера
* Алгоритм [DEFLATE](http://ru.wikipedia.org/wiki/DEFLATE) — популярный свободный алгоритм сжатия (используется в библиотеке zlib)
* [Дельта-кодирование](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D1%8C%D1%82%D0%B0-%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — эффективно для сжатия данных с часто повторяющимися последовательностями
* [Инкрементное кодирование](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — дельта-кодирование, применяемое к последовательности строк
* Семейство алгоритмов словарного сжатия Лемпеля — Зива:
  + [LZ77](http://ru.wikipedia.org/wiki/LZ77) — родоначальник семейства LZ77-алгоритмов
    - [LZ77-PM](http://ru.wikipedia.org/w/index.php?title=LZ77-PM&action=edit&redlink=1)
    - [LZFG](http://ru.wikipedia.org/w/index.php?title=LZFG&action=edit&redlink=1)
      * [LZFG-PM](http://ru.wikipedia.org/w/index.php?title=LZFG-PM&action=edit&redlink=1)
    - [LZP](http://ru.wikipedia.org/w/index.php?title=LZP&action=edit&redlink=1)
    - [LZBW](http://ru.wikipedia.org/w/index.php?title=LZBW&action=edit&redlink=1)
    - [LZSS](http://ru.wikipedia.org/w/index.php?title=LZSS&action=edit&redlink=1)
      * [LZB](http://ru.wikipedia.org/w/index.php?title=LZB&action=edit&redlink=1)
      * [LZH](http://ru.wikipedia.org/wiki/LZH)
      * [LZRW1](http://ru.wikipedia.org/w/index.php?title=LZRW1&action=edit&redlink=1)
  + [LZ78](http://ru.wikipedia.org/wiki/LZ78) — родоначальник семейства LZ78 алгоритмов
    - [Алгоритм Лемпеля — Зива — Велча](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9B%D0%B5%D0%BC%D0%BF%D0%B5%D0%BB%D1%8F_%E2%80%94_%D0%97%D0%B8%D0%B2%D0%B0_%E2%80%94_%D0%92%D0%B5%D0%BB%D1%87%D0%B0) (также известен как [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *LZW*) — сжатие без потерь
      * [LZW-PM](http://ru.wikipedia.org/w/index.php?title=LZW-PM&action=edit&redlink=1)
    - [LZMW](http://ru.wikipedia.org/w/index.php?title=LZMW&action=edit&redlink=1)
  + [LZMA](http://ru.wikipedia.org/wiki/LZMA) — сокращение от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Lempel-Ziv-Markov chain-Algorithm*
  + [LZO](http://ru.wikipedia.org/wiki/LZO) — алгоритм сжатия, ориентированный на скорость
* [Алгоритм сжатия PPM](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F_PPM)
* [Кодирование длин серий](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B4%D0%BB%D0%B8%D0%BD_%D1%81%D0%B5%D1%80%D0%B8%D0%B9) (Групповое кодирование, также известен как [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *RLE*) — последовательная серия одинаковых элементов заменяется на два символа: элемент и число его повторений
* [Алгоритм SEQUITUR](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_SEQUITUR&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/SEQUITUR_algorithm)) — сжатие без потерь, автоматическое адаптивное построение контекстно-свободной грамматики для обрабатываемых данных
* [Вейвлет-кодирование на основе вложенных нуль-деревьев](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B5%D0%B9%D0%B2%D0%BB%D0%B5%D1%82-%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5_%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D0%BD%D1%83%D0%BB%D1%8C-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D0%B5%D0%B2&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Embedded_Zerotree_Wavelet)) (EZW-кодирование)
* [Энтропийное кодирование](http://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — схема кодирования, которая присваивает коды символам таким образом, чтобы соотнести длину кодов с вероятностью появления символов
  + [Алгоритм Шеннона — Фано](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A8%D0%B5%D0%BD%D0%BD%D0%BE%D0%BD%D0%B0_%E2%80%94_%D0%A4%D0%B0%D0%BD%D0%BE) — самый простой алгоритм кодирования
  + [Алгоритм Хаффмана](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0) — алгоритм построения кода при помощи кодовых деревьев
    - [Адаптивное кодирование Хаффмана](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Adaptive_Huffman_coding)) — техника адаптивного кодирования, основывающаяся на коде Хаффмана
  + [Усечённое двоичное кодирование](http://ru.wikipedia.org/w/index.php?title=%D0%A3%D1%81%D0%B5%D1%87%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Truncated_binary_encoding)) — используется для однородного вероятностного распределения с конечным алфавитом
  + [Арифметическое кодирование](http://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — развитие энтропийного кодирования
    - [Адаптивное арифметическое кодирование](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) — техника адаптивного кодирования, основывающаяся на арифметическом кодировании
  + [Кодирование расстояний](http://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Range_encoding)) — метод сжатия данных, который близок по эффективности к арифметическому кодированию
* [Энтропийное кодирование с известными характеристиками](http://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Унарное кодирование](http://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — код, который представляет число nв виде nединиц с замыкающим нулём
  + [дельта](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D1%8C%D1%82%D0%B0-%D0%BA%D0%BE%D0%B4_%D0%AD%D0%BB%D0%B8%D0%B0%D1%81%D0%B0)|[гамма](http://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B0-%D0%BA%D0%BE%D0%B4_%D0%AD%D0%BB%D0%B8%D0%B0%D1%81%D0%B0)|[омега](http://ru.wikipedia.org/wiki/%D0%9E%D0%BC%D0%B5%D0%B3%D0%B0-%D0%BA%D0%BE%D0%B4_%D0%AD%D0%BB%D0%B8%D0%B0%D1%81%D0%B0)-кодирование Элиаса ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Elias coding*) — универсальный код, кодирующий положительные целые числа
  + [Кодирование Фибоначчи](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8) — универсальный код, который кодирует положительные целые числа в двоичные кодовые слова
  + [Кодирование Голомба](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%93%D0%BE%D0%BB%D0%BE%D0%BC%D0%B1%D0%B0) — форма энтропийного кодирования, которая оптимальна для алфавитов с геометрическим распределением
  + [Кодирование Райса](http://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A0%D0%B0%D0%B9%D1%81%D0%B0&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Rice_coding)) — форма энтропийного кодирования, которая оптимальна для алфавитов с геометрическим распределением

1. **Сеть Internet, доменная организация, семейство протоколов TCP/IP.**

**Интерне́т** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Internet*, МФА: [[ˈɪn.tə.net]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82)[[1]](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82#cite_note-1)) — всемирная система объединённых [компьютерных сетей](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Часто упоминается как **Всемирная сеть** и **Глобальная сеть**, а также просто **Сеть**[[2]](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82#cite_note-2). Построена на базе [стека протоколов TCP/IP](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%B2_TCP/IP). На основе Интернета работает [Всемирная паутина](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0) (World Wide Web, WWW) и множество других [систем передачи данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

## Ключевые принципы

Интернет состоит из многих тысяч корпоративных, научных, правительственных и домашних [компьютерных](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) сетей. Объединение сетей разной архитектуры и [топологии](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D1%82%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) стало возможно благодаря протоколу [IP](https://ru.wikipedia.org/wiki/IP) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Internet Protocol*) и принципу маршрутизации [пакетов данных](https://ru.wikipedia.org/wiki/IP-%D0%BF%D0%B0%D0%BA%D0%B5%D1%82).

Протокол IP был специально создан агностическим в отношении физических каналов связи. То есть любая система (сеть) передачи цифровых данных, проводная или беспроводная, для которой существует стандарт [инкапсуляции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%28%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8%29) в неё IP-пакетов, может передавать и трафик Интернета. Агностицизм протокола IP, в частности, означает, что компьютер или маршрутизатор должен знать тип сетей, к которым он непосредственно присоединён, и уметь работать с этими сетями; но не обязан (и в большинстве случаев не может) знать, какие сети находятся за маршрутизаторами.

На стыках сетей специальные [маршрутизаторы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80) (программные или аппаратные) занимаются автоматической сортировкой и перенаправлением пакетов данных, исходя из [IP-адресов](https://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81) получателей этих пакетов. Протокол IP образует единое адресное пространство в масштабах всего мира, но в каждой отдельной сети может существовать и собственное адресное подпространство, которое выбирается исходя из [класса сети](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D1%8B_%D1%81%D0%B5%D1%82%D0%B5%D0%B9&action=edit&redlink=1). Такая организация IP-адресов позволяет маршрутизаторам однозначно определять дальнейшее направление для каждого пакета данных. В результате между отдельными сетями Интернета не возникает конфликтов, и данные беспрепятственно и точно передаются из сети в сеть по всей планете и ближнему космосу.

Сам протокол IP был рождён в [дискуссиях](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%83%D1%81%D1%81%D0%B8%D1%8F) внутри организации [IETF](https://ru.wikipedia.org/wiki/IETF) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Internet Engineering Task Force*; Task force — группа специалистов для решения конкретной задачи), чьё название можно вольно перевести как «Группа по решению задач проектирования Интернета». IETF и её рабочие группы по сей день занимаются развитием протоколов Всемирной сети. IETF открыта для публичного участия и обсуждения. Комитеты организации публикуют так называемые документы [RFC](https://ru.wikipedia.org/wiki/RFC). В этих документах даются технические спецификации и точные объяснения по многим вопросам. Некоторые документы RFC возводятся организацией [IAB](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B2%D0%B5%D1%82_%D0%BF%D0%BE_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B5_%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%D0%B0) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Internet Architecture Board* — Совет по архитектуре Интернета) в статус стандартов Интернета ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Internet Standard*). С [1992 года](https://ru.wikipedia.org/wiki/1992_%D0%B3%D0%BE%D0%B4) IETF, IAB и ряд других интернет-организаций входят в [Общество Интернета](https://ru.wikipedia.org/w/index.php?title=ISOCKS&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Internet Society, ISOC*). Общество Интернета предоставляет организационную основу для разных исследовательских и консультативных групп, занимающихся развитием Интернета.

## Структура (сервисы и услуги)

В настоящее время в Интернете существует достаточно большое количество [сервисов](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D1%83%D0%B3%D0%B0), обеспечивающих работу со всем спектром ресурсов. Наиболее известными среди них являются:

* **сервис** [**DNS**](https://ru.wikipedia.org/wiki/DNS), или система доменных имен, обеспечивающий возможность использования для адресации узлов сети мнемонических имен вместо числовых адресов;
* [**электронная почта**](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0) ([E-mail](https://ru.wikipedia.org/wiki/E-mail)), обеспечивающая возможность обмена сообщениями одного человека с одним или несколькими абонентами;
* **сервис** [**IRC**](https://ru.wikipedia.org/wiki/IRC), предназначенный для поддержки текстового общения в реальном времени (chat);
* [**телеконференции**](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%8F), или группы новостей ([Usenet](https://ru.wikipedia.org/wiki/Usenet" \o "Usenet)), обеспечивающие возможность коллективного обмена сообщениями;
* **сервис** [**FTP**](https://ru.wikipedia.org/wiki/FTP) — система файловых архивов, обеспечивающая хранение и пересылку файлов различных типов;
* **сервис** [**Telnet**](https://ru.wikipedia.org/wiki/Telnet), предназначенный для управления удаленными компьютерами в терминальном режиме;
* [**World Wide Web**](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0) (WWW, W3, «Всемирная паутина») — гипертекстовая (гипермедиа) система, предназначенная для интеграции различных сетевых ресурсов в единое информационное пространство;
* [**Потоковое мультимедиа**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BC%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0).

Перечисленные выше сервисы относятся к стандартным. Это означает, что принципы построения клиентского и серверного программного обеспечения, а также протоколы взаимодействия сформулированы в виде международных стандартов. Следовательно, разработчики программного обеспечения при практической реализации обязаны выдерживать общие технические требования.

Наряду со стандартными сервисами существуют и нестандартные, представляющие собой оригинальную разработку той или иной компании. В качестве примера можно привести различные системы типа Instant Messenger (своеобразные интернет-пейджеры — ICQ, AOl, Demos on-line и т. п.), системы интернет-телефонии, трансляции радио и видео и т. д. Важной особенностью таких систем является отсутствие международных стандартов, что может привести к возникновению технических конфликтов с другими подобными сервисами.

Для стандартных сервисов также стандартизируется и интерфейс взаимодействия с протоколами транспортного уровня. В частности, за каждым программным сервером резервируются стандартные номера TCP- и UDP-портов, которые остаются неизменными независимо от особенностей той или иной фирменной реализации как компонентов сервиса, так и транспортных протоколов. Номера портов клиентского программного обеспечения так жестко не регламентируются. Это объясняется следующими факторами:

* во-первых, на пользовательском узле может функционировать несколько копий клиентской программы, и каждая из них должна однозначно идентифицироваться транспортным протоколом, то есть за каждой копией должен быть закреплен свой уникальный номер порта;
* во-вторых, клиенту важна регламентация портов сервера, чтобы знать, куда направлять запрос, а сервер сможет ответить клиенту, узнав адрес из поступившего запроса.

Услуги

Сейчас наиболее популярные [услуги Интернета](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81) — это:

|  |  |
| --- | --- |
| * [Всемирная паутина](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0)   + [Веб-форумы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%84%D0%BE%D1%80%D1%83%D0%BC)   + [Блоги](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%B3)   + [Вики](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8)-проекты (и, в частности, [Википедия](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F))   + [Интернет-магазины](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD)   + [Интернет-аукционы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%B0%D1%83%D0%BA%D1%86%D0%B8%D0%BE%D0%BD)   + [Социальные сети](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_%28%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%29) * [Электронная почта](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0) и [списки рассылки](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D0%BF%D0%BE%D1%87%D1%82%D1%8B) * [Группы новостей](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%BF%D0%BF%D1%8B_%D0%BD%D0%BE%D0%B2%D0%BE%D1%81%D1%82%D0%B5%D0%B9) (в основном, [Usenet](https://ru.wikipedia.org/wiki/Usenet)) * [Файлообменные сети](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB%D0%BE%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8) * [Электронные платёжные системы](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%91%D0%B6%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) | * [Интернет-радио](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%80%D0%B0%D0%B4%D0%B8%D0%BE) * [Интернет-телевидение](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%82%D0%B5%D0%BB%D0%B5%D0%B2%D0%B8%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5) [IPTV](https://ru.wikipedia.org/wiki/IPTV) * [IP-телефония](https://ru.wikipedia.org/wiki/IP-%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%B8%D1%8F) * [Мессенджеры](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%81%D1%81%D0%B5%D0%BD%D0%B4%D0%B6%D0%B5%D1%80) * [FTP-серверы](https://ru.wikipedia.org/wiki/FTP-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) * [IRC](https://ru.wikipedia.org/wiki/IRC) (реализовано также как [веб-чаты](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%87%D0%B0%D1%82%D1%8B)) * [Поисковые системы](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) * [Интернет-реклама](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%80%D0%B5%D0%BA%D0%BB%D0%B0%D0%BC%D0%B0) * [Удалённые терминалы](https://ru.wikipedia.org/w/index.php?title=%D0%A3%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B0%D0%BB&action=edit&redlink=1) * [Удалённое управление](https://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) * [Многопользовательские игры](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) * [Веб 2.0](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1_2.0) |

**Доменная структура**

Доменную структуру сети можно назвать цивилизованным способом организации работы локальной сети, к которому должен стремиться каждый администратор, создающий локальные сети.

Данный способ подразумевает наличие в локальной сети специализированного, выделенного компьютера – сервера, который занимается исключительно обслуживанием работы сети. В идеальном случае количество управляющих серверов не меньше двух, что позволяет обеспечить функционирование сети в случае выхода из строя основного сервера.

Управляющий сервер называется *контроллером домена.* Его единственная задача – управление сетью. Часто встречается ситуация, когда контроллер домена решает дополнительные задач, например выступает в качестве файлового сервера. Подобное положение вещей является крайне нежелательным, поскольку не только усложняет обслуживание сервера, но и делает сеть более подверженной риску выхода сервера из строя.

Под нужды будущего контроллера домена выделяется мощный компьютер, обладающий производительной дисковой подсистемой, мощным процессором и большим объемом оперативной памяти. На него устанавливается серверная операционная система, специально предназначенная для таких случаев.

Как уже упоминалось ранее, серверная операционная система содержит в себе целый ряд инструментов, с помощью которых осуществляется администрирование локальной сети. К ним относятся Active Directory, DNS– и DHCP-сервер, хранилище сертификатов и т. д.

Использование доменной структуры имеет ряд преимуществ, среди которых:

           контролируемое подключение к локальной сети путем использования учетной записи пользователя;

           полный контроль над участниками сети;

           мощная система управления правами доступа;

           контролируемая организация доступа к общим ресурсам;

           система архивирования;

           настраиваемые политики работы в локальной сети;

           автоматическая установка необходимых пакетов обновления системы и программных продуктов;

           корпоративная антивирусная защита локальной сети.

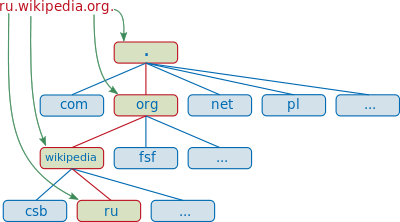
Поскольку контроллер домена является центральным компонентом локальной сети, от которого зависит работа всей локальной сети, обеспечение его работоспособности – важнейшая задача системного администратора. Именно он должен позаботиться о том, что бы настроить резервный сервер, который сможет выполнять все функции контроллера домена в случае, если с ним что-то произойдет. При наличии резервного компьютера он получает название вторичного контроллера домена, а основной сервер – первичного контроллера домена или основного контроллера домена.

**DNS** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Domain Name System* — система доменных имён) — компьютерная [распределённая система](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) для получения информации о [доменах](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BC%D1%8F). Чаще всего используется для получения IP-адреса по имени [хоста](http://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82) (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене ([SRV-запись](http://ru.wikipedia.org/wiki/SRV-%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C)).

[Распределённая база данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) DNS поддерживается с помощью иерархии [DNS-серверов](http://ru.wikipedia.org/wiki/DNS-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), взаимодействующих по определённому [протоколу](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Основой DNS является представление об иерархической структуре [доменного имени](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BC%D1%8F) и *зонах*. Каждый сервер, отвечающий за имя, может *делегировать* ответственность за дальнейшую часть домена другому серверу (с административной точки зрения — другой организации или человеку), что позволяет возложить ответственность за актуальность информации на серверы различных организаций (людей), отвечающих только за «свою» часть доменного имени.

Начиная с 2010 года, в систему DNS внедряются средства проверки целостности передаваемых данных, называемые *DNS Security Extensions* ([DNSSEC](http://ru.wikipedia.org/wiki/DNSSEC)). Передаваемые данные не шифруются, но их достоверность проверяется криптографическими способами. Внедряемый стандарт [DANE](http://ru.wikipedia.org/wiki/DANE) обеспечивает передачу средствами DNS достоверной криптографической информации ([сертификатов](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D1%8E%D1%87%D0%B0)), используемых для установления безопасных и защищённых соединений [транспортного](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) и [прикладного](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) уровней.

[](http://commons.wikimedia.org/wiki/File:DNS-names-ru.svg?uselang=ru)

[http://bits.wikimedia.org/static-1.22wmf8/skins/common/images/magnify-clip.png](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:DNS-names-ru.svg)

Пример структуры доменного имени

## Ключевые характеристики DNS

DNS обладает следующими характеристиками:

* *Распределённость администрирования*. *Ответственность* за разные части иерархической структуры несут разные люди или организации.
* *Распределённость хранения информации*. Каждый узел сети в обязательном порядке должен хранить только те данные, которые входят в его *зону ответственности* и (возможно) адреса *корневых DNS-серверов*.
* [*Кеширование*](http://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D1%88) *информации*. Узел *может* хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть.
* *Иерархическая структура*, в которой все узлы объединены в [дерево](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%28%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29), и каждый узел может или самостоятельно определять работу нижестоящих узлов, или *делегировать* (передавать) их другим узлам.
* *Резервирование*. За хранение и обслуживание своих узлов (зон) отвечают (обычно) несколько серверов, разделённые как физически, так и логически, что обеспечивает сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

DNS важна для работы [Интернета](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), так как для соединения с узлом необходима информация о его [IP-адресе](http://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81), а для людей проще запоминать буквенные (обычно осмысленные) адреса, чем последовательность цифр IP-адреса. В некоторых случаях это позволяет использовать виртуальные серверы, например, HTTP-серверы, различая их по имени запроса. Первоначально преобразование между доменными и IP-адресами производилось с использованием специального [текстового файла](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) [hosts](http://ru.wikipedia.org/wiki/Hosts), который составлялся централизованно и автоматически рассылался на каждую из машин в своей локальной сети. С ростом Сети возникла необходимость в эффективном, автоматизированном механизме, которым и стала DNS.

## Терминология и принципы работы

Ключевыми понятиями DNS являются:

* [**Доме́н**](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BC%D1%8F) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *domain* — область) — узел в дереве имён, вместе со всеми подчинёнными ему узлами (если таковые имеются), то есть именованная *ветвь* или *поддерево* в дереве имен. Структура доменного имени отражает порядок следования узлов в иерархии; доменное имя читается слева направо от младших доменов к доменам высшего уровня (в порядке повышения значимости), корневым доменом всей системы является точка ('.'), ниже идут домены первого уровня (географические или тематические), затем — домены второго уровня, третьего и т. д. (например, для адреса ru.wikipedia.org домен первого уровня — org, второго wikipedia, третьего ru). На практике точку в конце имени часто опускают, но она бывает важна в случаях разделения между относительными доменами и [FQDN](http://ru.wikipedia.org/wiki/FQDN) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Fully Qualifed Domain Name*, полностью определённое имя домена).
* [**Поддомен**](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%B4%D0%BE%D0%BC%D0%B5%D0%BD) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *subdomain*) — подчинённый домен (например, wikipedia.org — поддомен домена org, а ru.wikipedia.org — домена wikipedia.org). Теоретически такое деление может достигать глубины 127 уровней, а каждая метка может содержать до 63 символов, пока общая длина вместе с точками не достигнет 254 символов. Но на практике [регистраторы доменных имён](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%B4%D0%BE%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D0%B8%D0%BC%D1%91%D0%BD) используют более строгие ограничения. Например, если у вас есть домен вида mydomain.ru, вы можете создать для него различные поддомены вида *mysite1.mydomain.ru*, *mysite2.mydomain.ru* и т. д.
* [**Ресурсная запись**](http://ru.wikipedia.org/wiki/DNS#.D0.97.D0.B0.D0.BF.D0.B8.D1.81.D0.B8_DNS) — единица хранения и передачи информации в DNS. Каждая ресурсная запись имеет *имя* (то есть привязана к определенному **Доменному имени**, узлу в дереве имен), *тип* и *поле данных*, формат и содержание которого зависит от *типа*.
* **Зона** — часть дерева доменных имен (включая **ресурсные записи**), размещаемая как единое целое на некотором сервере доменных имен (**DNS-сервере**, см. ниже), а чаще — одновременно на нескольких серверах (см. ниже). Целью выделения части дерева в отдельную зону является передача **ответственности** (см. ниже) за соответствующий **домен** другому лицу или организации. Это называется **делегированием** (см. ниже). Как связная часть дерева, **зона** внутри тоже представляет собой дерево. Если рассматривать пространство имен DNS как структуру из зон, а не отдельных узлов/имен, тоже получается дерево; оправданно говорить о родительских и дочерних зонах, о старших и подчиненных. На практике, большинство зон 0-го и 1-го уровня ('.', ru, com, …) состоят из единственного узла, которому непосредственно подчиняются дочерние зоны. В больших корпоративных доменах (2-го и более уровней) иногда встречается образование дополнительных подчиненных уровней без выделения их в дочерние зоны.
* **Делегирование** — операция передачи **ответственности** за часть дерева доменных имен другому лицу или организации. За счет делегирования в DNS обеспечивается распределенность администрирования и хранения. Технически **делегирование** выражается в выделении этой части дерева в отдельную **зону**, и размещении этой **зоны** на **DNS-сервере** (см. ниже), управляемом этим лицом или организацией. При этом в родительскую зону включаются «[склеивающие](http://ru.wikipedia.org/wiki/Glue)» **ресурсные записи** (NS и А), содержащие указатели на **DNS-сервера** дочерней зоны, а вся остальная информация, относящаяся к дочерней зоне, хранится уже на **DNS-серверах** дочерней зоны.
* [**DNS-сервер**](http://ru.wikipedia.org/wiki/DNS-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) — специализированное ПО для обслуживания DNS, а также компьютер, на котором это ПО выполняется. DNS-сервер может быть ответственным за некоторые зоны и/или может перенаправлять запросы вышестоящим серверам.
* [**DNS-клиент**](http://ru.wikipedia.org/wiki/DNS-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82) — специализированная библиотека (или программа) для работы с DNS. В ряде случаев DNS-сервер выступает в роли DNS-клиента.
* **Авторитетность** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *authoritative*) — признак размещения зоны на DNS-сервере. Ответы DNS-сервера могут быть двух типов: *авторитетные* (когда сервер заявляет, что сам отвечает за зону) и *неавторитетные* ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Non-authoritative*), когда сервер обрабатывает запрос, и возвращает ответ других серверов. В некоторых случаях вместо передачи запроса дальше DNS-сервер может вернуть уже известное ему (по запросам ранее) значение (режим кеширования).
* [**DNS-запрос**](http://ru.wikipedia.org/w/index.php?title=DNS-%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81&action=edit&redlink=1) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *DNS query*) — запрос от клиента (или сервера) серверу. Запрос может быть *рекурсивным* или *нерекурсивным* (см. [Рекурсия](http://ru.wikipedia.org/wiki/DNS#.D0.A0.D0.B5.D0.BA.D1.83.D1.80.D1.81.D0.B8.D1.8F)).

Система DNS содержит иерархию **DNS-серверов**, соответствующую иерархии **зон**. Каждая **зона** поддерживается как минимум одним *авторитетным сервером DNS* (от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *authoritative* — авторитетный), на котором расположена информация о домене.

Имя и IP-адрес не [тождественны](http://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B6%D0%B4%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F%29) — один IP-адрес может иметь множество имён, что позволяет поддерживать на одном компьютере множество [веб-сайтов](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82) (это называется [виртуальный хостинг](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%85%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3)). Обратное тоже справедливо — одному имени может быть сопоставлено множество IP-адресов: это позволяет создавать [балансировку нагрузки](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BD%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8).

Для повышения устойчивости системы используется множество серверов, содержащих идентичную информацию, а в протоколе есть средства, позволяющие поддерживать синхронность информации, расположенной на разных серверах. Существует 13 [корневых серверов](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BD%D0%B5%D0%B2%D1%8B%D0%B5_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D1%8B_DNS), их адреса практически не изменяются.[[1]](http://ru.wikipedia.org/wiki/DNS#cite_note-1)

Протокол DNS использует для работы [TCP](http://ru.wikipedia.org/wiki/TCP)- или [UDP](http://ru.wikipedia.org/wiki/UDP)-[порт](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82_%28TCP/UDP%29) 53 для ответов на запросы. Традиционно запросы и ответы отправляются в виде одной UDP [датаграммы](http://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). TCP используется для [AXFR](http://ru.wikipedia.org/wiki/AXFR)-запросов.

### Рекурсия

Термином **Рекурсия** в DNS обозначают алгоритм поведения **DNS-сервера**, при котором сервер выполняет от имени клиента полный поиск нужной информации во всей системе DNS, при необходимости обращаясь к другим **DNS-серверам**.

**DNS-запрос** может быть *рекурсивным* — требующим полного поиска, — и *нерекурсивным* (или *итеративным*) — не требующим полного поиска.

Аналогично, **DNS-сервер** может быть *рекурсивным* (умеющим выполнять полный поиск) и *нерекурсивным* (не умеющим выполнять полный поиск). Некоторые программы DNS-серверов, например, [BIND](http://ru.wikipedia.org/wiki/BIND), можно сконфигурировать так, чтобы запросы одних клиентов выполнялись *рекурсивно*, а запросы других — *нерекурсивно*.

При ответе на *нерекурсивный* запрос, а также — при неумении или запрете выполнять *рекурсивные* запросы, — DNS-сервер либо возвращает данные о зоне, за которую он **ответствен**, либо возвращает адреса серверов, которые обладают большим объёмом информации о запрошенной зоне, чем отвечающий сервер, чаще всего — адреса корневых серверов.

В случае *рекурсивного* запроса **DNS-сервер** опрашивает серверы (в порядке убывания уровня зон в имени), пока не найдёт ответ или не обнаружит, что домен не существует. (На практике поиск начинается с наиболее близких к искомому DNS-серверов, если информация о них есть в кэше и не устарела, сервер может не запрашивать другие DNS-серверы.)

Рассмотрим на примере работу всей системы.

Предположим, мы набрали в [браузере](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) адрес ru.wikipedia.org. Браузер спрашивает у сервера DNS: «какой IP-адрес у ru.wikipedia.org»? Однако, сервер DNS может ничего не знать не только о запрошенном имени, но даже обо всём домене wikipedia.org. В этом случае сервер обращается к [*корневому серверу*](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BD%D0%B5%D0%B2%D1%8B%D0%B5_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D1%8B_DNS) — например, 198.41.0.4. Этот сервер сообщает — «У меня нет информации о данном адресе, но я знаю, что 204.74.112.1 является ответственным за зону org.» Тогда сервер DNS направляет свой запрос к 204.74.112.1, но тот отвечает «У меня нет информации о данном сервере, но я знаю, что 207.142.131.234 является ответственным за зону wikipedia.org.» Наконец, тот же запрос отправляется к третьему DNS-серверу и получает ответ — IP-адрес, который и передаётся клиенту — браузеру.

В данном случае при разрешении имени, то есть в процессе поиска IP по имени:

* браузер отправил известному ему DNS-серверу *рекурсивный* запрос — в ответ на такой тип запроса сервер обязан вернуть «готовый результат», то есть IP-адрес, либо пустой ответ и код ошибки NXDOMAIN;
* DNS-сервер, получивший запрос от браузера, последовательно отправлял *нерекурсивные* запросы, на которые получал от других DNS-серверов ответы, пока не получил ответ от сервера, ответственного за запрошенную зону;
* остальные упоминавшиеся DNS-серверы обрабатывали запросы *нерекурсивно* (и, скорее всего, не стали бы обрабатывать запросы рекурсивно, даже если бы такое требование стояло в запросе).

Иногда допускается, чтобы запрошенный сервер передавал *рекурсивный* запрос «вышестоящему» DNS-серверу и дожидался готового ответа.

При *рекурсивной* обработке запросов все ответы проходят через DNS-сервер, и он получает возможность [*кэшировать*](http://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88) их. Повторный запрос на те же имена обычно не идет дальше *кэша* сервера, обращения к другим серверам не происходит вообще. Допустимое время хранения ответов в *кэше* приходит вместе с ответами (поле *TTL* [**ресурсной записи**](http://ru.wikipedia.org/wiki/DNS#.D0.97.D0.B0.D0.BF.D0.B8.D1.81.D0.B8_DNS)).

Рекурсивные запросы требуют больше ресурсов от сервера (и создают больше трафика), так что обычно принимаются от «известных» владельцу сервера узлов (например, провайдер предоставляет возможность делать рекурсивные запросы только своим клиентам, в корпоративной сети рекурсивные запросы принимаются только из локального сегмента). Нерекурсивные запросы обычно принимаются ото всех узлов сети (и содержательный ответ даётся только на запросы о зоне, которая размещена на узле, на DNS-запрос о других зонах обычно возвращаются адреса других серверов).

### Обратный DNS-запрос

Основная статья: [**Обратный запрос DNS**](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81_DNS)

DNS используется в первую очередь для преобразования символьных имён в IP-адреса, но он также может выполнять обратный процесс. Для этого используются уже имеющиеся средства DNS. Дело в том, что с записью DNS могут быть сопоставлены различные данные, в том числе и какое-либо символьное имя. Существует специальный домен in-addr.arpa, записи в котором используются для преобразования IP-адресов в символьные имена. Например, для получения DNS-имени для адреса 11.22.33.44 можно запросить у DNS-сервера запись 44.33.22.11.in-addr.arpa, и тот вернёт соответствующее символьное имя. Обратный порядок записи частей IP-адреса объясняется тем, что в IP-адресах старшие биты расположены в начале, а в символьных DNS-именах старшие (находящиеся ближе к корню) части расположены в конце.

## Записи DNS

Основная статья: [**Ресурсные записи DNS**](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BD%D1%8B%D0%B5_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%B8_DNS)

**Записи DNS**, или **Ресурсные записи** (*англ.* Resource Records, RR) — единицы хранения и передачи информации в DNS. Каждая ресурсная запись состоит из следующих полей:

* *имя* (NAME) — доменное имя, к которому привязана или которому «принадлежит» данная ресурсная запись,
* *TTL* (Time To Live) — допустимое время хранения данной ресурсной записи в кэше *неответственного* **DNS-сервера**,
* *тип* (TYPE) ресурсной записи — определяет формат и назначение данной ресурсной записи,
* *класс* (CLASS) ресурсной записи; ***теоретически*** считается, что DNS может использоваться не только с [TCP/IP](http://ru.wikipedia.org/wiki/TCP/IP), но и с другими типами сетей, код в поле *класс* определяет тип сети [[2]](http://ru.wikipedia.org/wiki/DNS#cite_note-2),
* длина поля данных (RDLEN),
* *поле данных* (RDATA), формат и содержание которого зависит от *типа* записи.

Наиболее важные типы DNS-записей:

* **Запись A** (*address record*) или **запись адреса** связывает имя хоста с адресом IP. Например, запрос A-записи на имя referrals.icann.org вернет его IP адрес — 192.0.34.164
* **Запись AAAA** (*IPv6 address record*) связывает имя хоста с адресом протокола [IPv6](http://ru.wikipedia.org/wiki/IPv6). Например, запрос AAAA-записи на имя K.ROOT-SERVERS.NET вернет его [IPv6](http://ru.wikipedia.org/wiki/IPv6) адрес — 2001:7fd::1
* **Запись CNAME** (*canonical name record*) или **каноническая запись имени** (псевдоним) используется для перенаправления на другое имя
* [**Запись MX**](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%B8%D1%81%D1%8C_MX) (*mail exchange*) или **почтовый обменник** указывает [сервер(ы) обмена почтой](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) для данного домена.
* **Запись NS** (*name server*) указывает на [DNS-сервер](http://ru.wikipedia.org/wiki/DNS-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) для данного домена.
* **Запись PTR** (*pointer*) или **запись указателя** связывает IP хоста с его каноническим именем. Запрос в домене in-addr.arpa на IP хоста в reverse форме вернёт имя ([FQDN](http://ru.wikipedia.org/wiki/FQDN)) данного хоста (см. [Обратный DNS-запрос](http://ru.wikipedia.org/wiki/DNS#.D0.9E.D0.B1.D1.80.D0.B0.D1.82.D0.BD.D1.8B.D0.B9_DNS-.D0.B7.D0.B0.D0.BF.D1.80.D0.BE.D1.81)). Например, (на момент написания), для IP адреса 192.0.34.164: запрос записи PTR 164.34.0.192.in-addr.arpa вернет его каноническое имя referrals.icann.org. В целях уменьшения объёма нежелательной корреспонденции (спама) многие серверы-получатели электронной почты могут проверять наличие PTR записи для хоста, с которого происходит отправка. В этом случае PTR запись для IP адреса должна соответствовать имени отправляющего почтового сервера, которым он представляется в процессе [SMTP](http://ru.wikipedia.org/wiki/SMTP)-сессии.
* **Запись** [**SOA**](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BD%D1%8B%D0%B5_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%B8_DNS) (*Start of Authority*) или **начальная запись зоны** указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, *тайминги* (параметры времени) кеширования зонной информации и взаимодействия DNS-серверов.
* [SRV-запись](http://ru.wikipedia.org/wiki/SRV-%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C) (*server selection*) указывает на серверы для сервисов, используется, в частности, для [Jabber](http://ru.wikipedia.org/wiki/Jabber) и [Active Directory](http://ru.wikipedia.org/wiki/Active_Directory).

**Стек протоколов TCP/IP** — набор [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) передачи данных, используемых в сетях, включая сеть [интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Название TCP/IP происходит из двух наиболее важных протоколов семейства — [Transmission Control Protocol](https://ru.wikipedia.org/wiki/Transmission_Control_Protocol) (TCP) и [Internet Protocol](https://ru.wikipedia.org/wiki/Internet_Protocol) (IP), которые были разработаны и описаны первыми в данном стандарте. Также изредка упоминается как [модель DOD](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_DOD) в связи с историческим происхождением от сети [ARPANET](https://ru.wikipedia.org/wiki/ARPANET) из 1970 годов (под управлением [DARPA](https://ru.wikipedia.org/wiki/DARPA), [Министерства обороны США](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D0%B5%D1%80%D1%81%D1%82%D0%B2%D0%BE_%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D0%BD%D1%8B_%D0%A1%D0%A8%D0%90))

Протоколы работают друг с другом в [стеке](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *stack*, стопка) — это означает, что протокол, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы [инкапсуляции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%28%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8%29). Например, протокол [TCP](https://ru.wikipedia.org/wiki/TCP) работает поверх протокола [IP](https://ru.wikipedia.org/wiki/IP).

Стек протоколов TCP/IP включает в себя четыре уровня:

* [прикладной уровень](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (application layer),
* [транспортный уровень](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (transport layer),
* [сетевой уровень](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (internet layer),
* [канальный уровень](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (link layer).

Протоколы этих уровней полностью реализуют функциональные возможности [модели OSI](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI). На стеке протоколов TCP/IP построено всё взаимодействие пользователей в IP-сетях. Стек является независимым от физической среды передачи данных.

## ровни стека TCP/IP

|  |  |  |
| --- | --- | --- |
| **Распределение протоколов по уровням модели TCP/IP** | | |
| 4 | **Прикладной** *«7 уровень»* | напр., [HTTP](https://ru.wikipedia.org/wiki/HTTP), [RTP](https://ru.wikipedia.org/wiki/RTP), [FTP](https://ru.wikipedia.org/wiki/FTP), [DNS](https://ru.wikipedia.org/wiki/DNS) |
| 3 | **Транспортный** | напр., [TCP](https://ru.wikipedia.org/wiki/TCP), [UDP](https://ru.wikipedia.org/wiki/UDP), [SCTP](https://ru.wikipedia.org/wiki/SCTP), [DCCP](https://ru.wikipedia.org/wiki/DCCP) *(*[*RIP*](https://ru.wikipedia.org/wiki/RIP2)*, протоколы маршрутизации, подобные* [*OSPF*](https://ru.wikipedia.org/wiki/OSPF)*, что работают поверх* [*IP*](https://ru.wikipedia.org/wiki/IP)*, являются частью сетевого уровня)* |
| 2 | **Сетевой** | Для TCP/IP это [IP](https://ru.wikipedia.org/wiki/IP) *(вспомогательные протоколы, вроде* [*ICMP*](https://ru.wikipedia.org/wiki/ICMP) *и* [*IGMP*](https://ru.wikipedia.org/wiki/IGMP)*, работают поверх IP, но тоже относятся к сетевому уровню; протокол* [*ARP*](https://ru.wikipedia.org/wiki/ARP) *является самостоятельным вспомогательным протоколом, работающим поверх канального уровня)* |
| 1 | **Канальный** | [Ethernet](https://ru.wikipedia.org/wiki/Ethernet), [IEEE 802.11](https://ru.wikipedia.org/wiki/IEEE_802.11) [Wireless Ethernet](https://ru.wikipedia.org/w/index.php?title=Wireless_Ethernet&action=edit&redlink=1), [SLIP](https://ru.wikipedia.org/wiki/SLIP), [Token Ring](https://ru.wikipedia.org/wiki/Token_Ring), [ATM](https://ru.wikipedia.org/wiki/ATM) и [MPLS](https://ru.wikipedia.org/wiki/MPLS), физическая среда и принципы кодирования информации, [T1](https://ru.wikipedia.org/wiki/T1), [E1](https://ru.wikipedia.org/wiki/E1_%28ISDN%29) |

### Прикладной уровень

### На [прикладном уровне](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) работает большинство сетевых приложений.

Эти программы имеют свои собственные протоколы обмена информацией, например, [HTTP](https://ru.wikipedia.org/wiki/HTTP) для [WWW](https://ru.wikipedia.org/wiki/WWW), [FTP](https://ru.wikipedia.org/wiki/FTP) (передача файлов), [SMTP](https://ru.wikipedia.org/wiki/SMTP) ([электронная почта](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0)), [SSH](https://ru.wikipedia.org/wiki/SSH) (безопасное соединение с удалённой машиной), [DNS](https://ru.wikipedia.org/wiki/DNS) (преобразование символьных имён в [IP-адреса](https://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81)) и многие другие.

В массе своей эти протоколы работают поверх [TCP](https://ru.wikipedia.org/wiki/TCP) или [UDP](https://ru.wikipedia.org/wiki/UDP) и привязаны к определённому порту, например:

* [HTTP](https://ru.wikipedia.org/wiki/HTTP) на TCP-порт 80 или 8080,
* [FTP](https://ru.wikipedia.org/wiki/FTP) на TCP-порт 20 (для передачи данных) и 21 (для управляющих команд),
* [SSH](https://ru.wikipedia.org/wiki/SSH) на TCP-порт 22,
* запросы [DNS](https://ru.wikipedia.org/wiki/DNS) на порт UDP (реже TCP) 53,
* обновление маршрутов по протоколу [RIP](https://ru.wikipedia.org/wiki/RIP2) на UDP-порт 520.

Эти порты определены Агентством по выделению имен и уникальных параметров протоколов ([IANA](https://ru.wikipedia.org/wiki/IANA)).

К этому уровню относятся: [Echo](https://ru.wikipedia.org/wiki/Echo), [Finger](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_finger), [Gopher](https://ru.wikipedia.org/wiki/Gopher_%28%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%29), [HTTP](https://ru.wikipedia.org/wiki/HTTP), [HTTPS](https://ru.wikipedia.org/wiki/HTTPS), [IMAP](https://ru.wikipedia.org/wiki/IMAP), [IMAPS](https://ru.wikipedia.org/w/index.php?title=IMAPS&action=edit&redlink=1), [IRC](https://ru.wikipedia.org/wiki/IRC), [NNTP](https://ru.wikipedia.org/wiki/NNTP), [NTP](https://ru.wikipedia.org/wiki/NTP), [POP3](https://ru.wikipedia.org/wiki/POP3), [POPS](https://ru.wikipedia.org/w/index.php?title=POPS&action=edit&redlink=1), [QOTD](https://ru.wikipedia.org/wiki/QOTD), [RTSP](https://ru.wikipedia.org/wiki/RTSP), [SNMP](https://ru.wikipedia.org/wiki/SNMP), [SSH](https://ru.wikipedia.org/wiki/SSH), [Telnet](https://ru.wikipedia.org/wiki/Telnet), [XDMCP](https://ru.wikipedia.org/wiki/XDMCP).

1. **Объектно-ориентированное программирование. Классы и объекты, наследование, интерфейсы.**
2. Технология разработки и сопровождения программ. Жизненный цикл программы.
3. Удаленный доступ к ресурсам сети.
4. Протоколы передачи файлов FTP и HTTP, язык разметки гипертекста HTML, разработка WEB-страниц, WWW-серверы.
5. CASE-средства и их использование при проектировании базы данных (БД).
6. Основные понятия технологии клиент-сервер.
7. Аппаратные и программные методы защиты данных и программ.

Литература.

1. Ахо, Сети Р., Ульман Дж. Компиляторы: принципы, техника реализации и инструменты. М., 2001.
2. Введение в криптографию / Под ред. В.В. Ященко. СПб.: МЦНМО, 2001.
3. Дейт К.Дж. Введение в системы баз данных. М.: Вильямс, 1999.
4. Дейтел Г. Введение в операционные системы. М.: Мир, 1987.
5. Кнут Д. Искусство программирования. Т. 1 - 3. М., СПб., Киев: ИД <Вильямс>, 2000.
6. Когаловский М.Р. Энциклопедия технологий баз данных. М.: Финансы и статистика, 2002.
7. Компьютерные сети. Учебный курс Microsoft Corporation, 1997.
8. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы, построение и анализ. М.: МЦНМО, 2000.
9. Котов В.Е., Сабельфельд В.К. Теория схем программ. М.: Наука, 1991.
10. Матфик С. Механизмы защиты в сетях ЭВМ. М.: Мир, 1993.
11. Мельников В.В. Защита информации в компьютерных системах. М.: Финансы и статистика, 1997.
12. Яблонский С.В. Введение в дискретную математику. М.: Наука, 2001.