# Тема лекции:

# Структуры данных

Структура данных – это способ хранения и организации данных, облегчающих доступ к этим данным и их модификацию.

Ни одна структура данных не является универсальной и не может подходить для всех целей, поэтому важно знать преимущества и ограничения, присущие им.

# ТИПИЧНЫЕ ОПЕРАЦИИ СО СТРУКТУРАМИ ДАННЫХ

Search

Insert

Delete

Minimum

Maximum

Sort

# СТРУКТУРА ДАННЫХ СТЕК

**Стек (stack) —** абстрактный тип данных, представляющий собой список элементов, организованных по принципу ***LIFO*** ( last in — first out, «последним пришёл — первым вышел»).

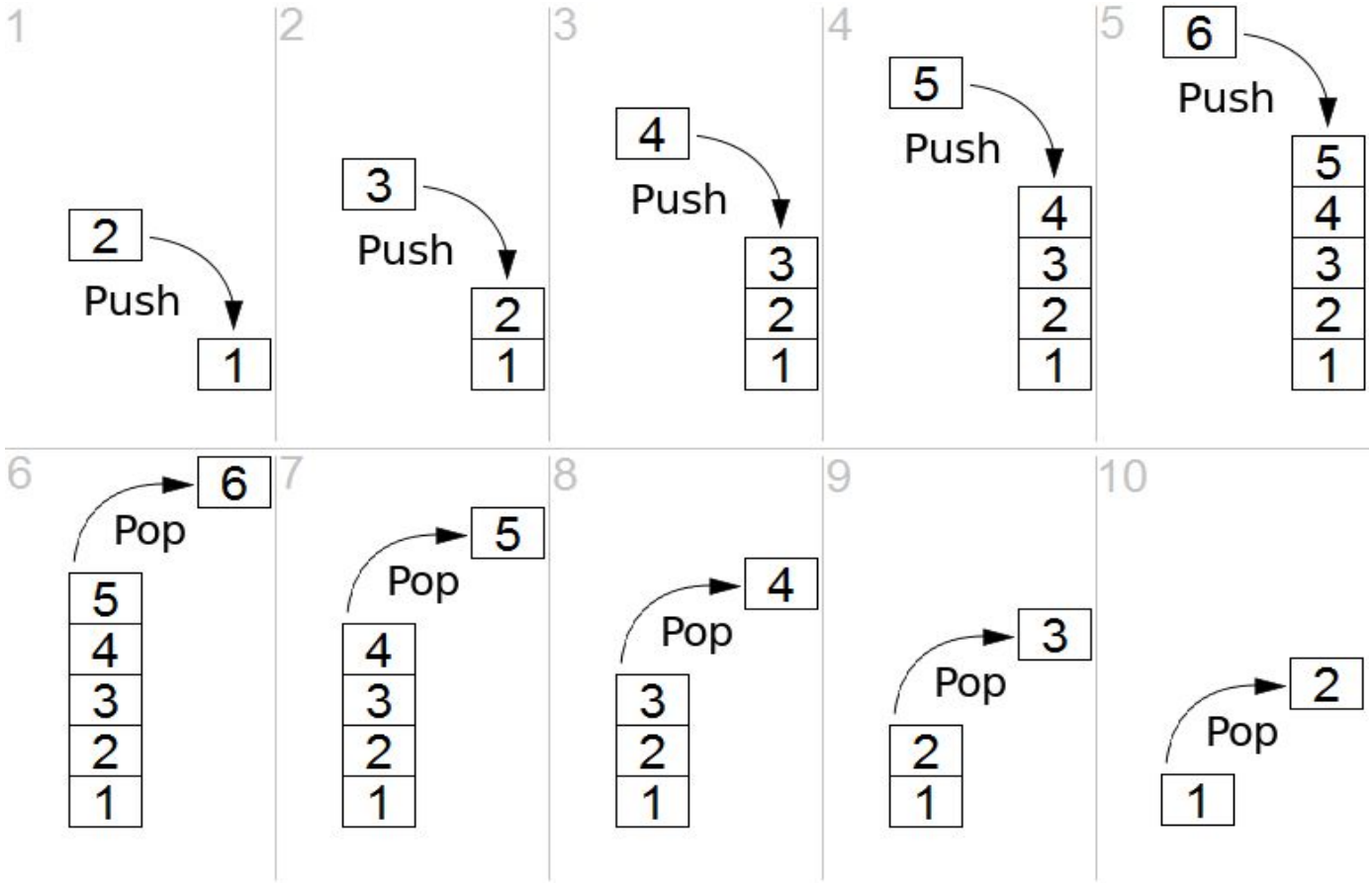
Зачастую стек реализуется в виде однонаправленного списка (каждый элемент в списке содержит помимо хранимой информации в стеке указатель на следующий элемент стека).

Но также часто стек располагается в одномерном массиве с упорядоченными адресами. При этом отпадает необходимость хранения в элементе стека явного указателя на следующий элемент стека, что экономит память. При этом указатель стека (***Stack Pointer***) обычно является регистром процессора и указывает на адрес головы стека.

**Регистр процессора** — блок ячеек памяти, образующий сверхбыструю оперативную память внутри процессора; используется самим процессором и большой частью недоступен программисту: например, при выборке из памяти очередной команды она помещается в регистр команд, к которому программист обратиться не может.

# СТРУКТУРА ДАННЫХ СТЕК

Возможны три операции со стеком: добавление элемента (иначе проталкивание, ***push***), удаление элемента (***pop***) и чтение головного элемента (***peek***).



# СВЯЗНЫЙ СПИСОК

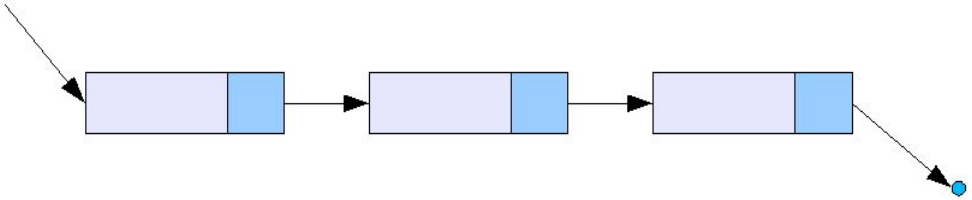
**Связный список** — базовая динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка.

Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

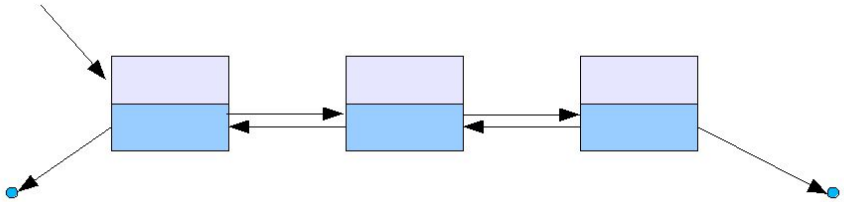
Существуют односвязные списки (одна ссылка на следующий элемент), двусвязные списки (ссылка на предыдущий и последующий элементы), кольцевые и др.

# СВЯЗНЫЙ СПИСОК

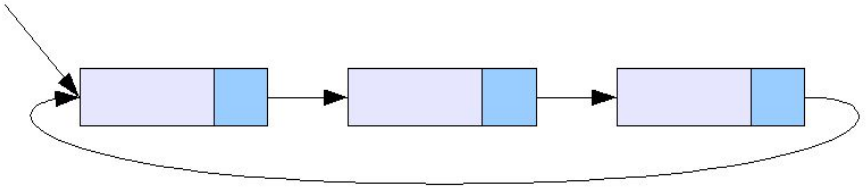
**Односвязный список**



**Двусвязный список**



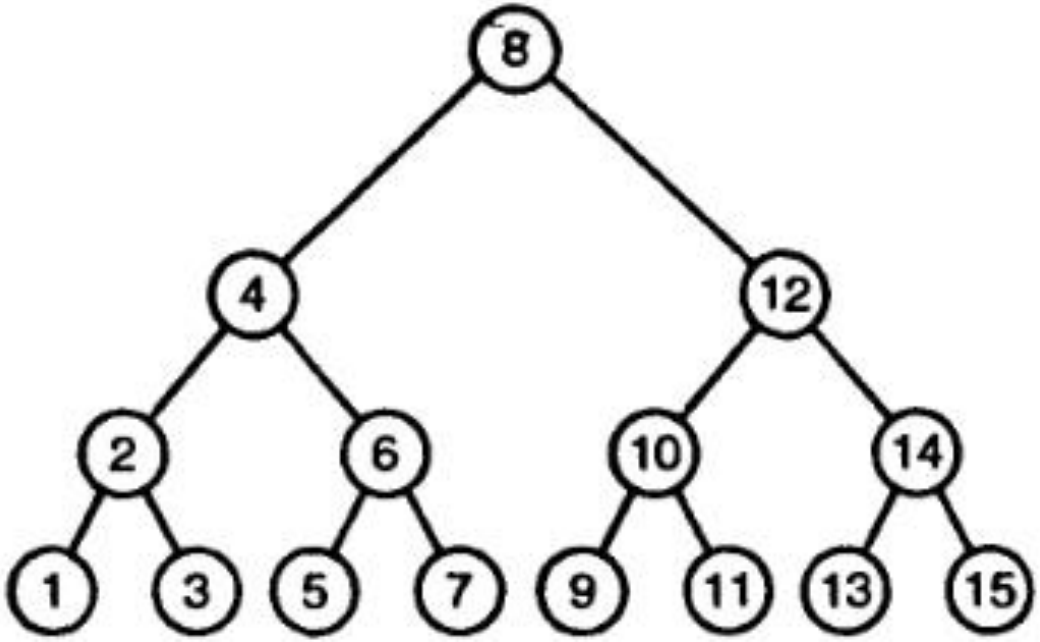
**Кольцевой связный список**

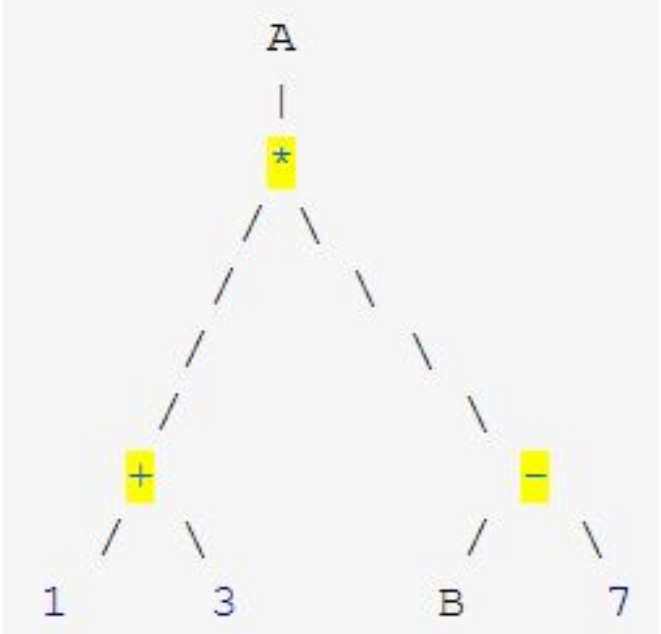


# БИНАРНОЕ ДЕРЕВО

**Двоичное дерево** — иерархическая структура данных, в которой каждый узел имеет не более двух потомков (детей). Как правило, первый называется родительским узлом, а дети называются левым и правым наследниками.

A = (1 + 3) * (B - 7)



# КРАСНО-ЧЕРНОЕ ДЕРЕВО

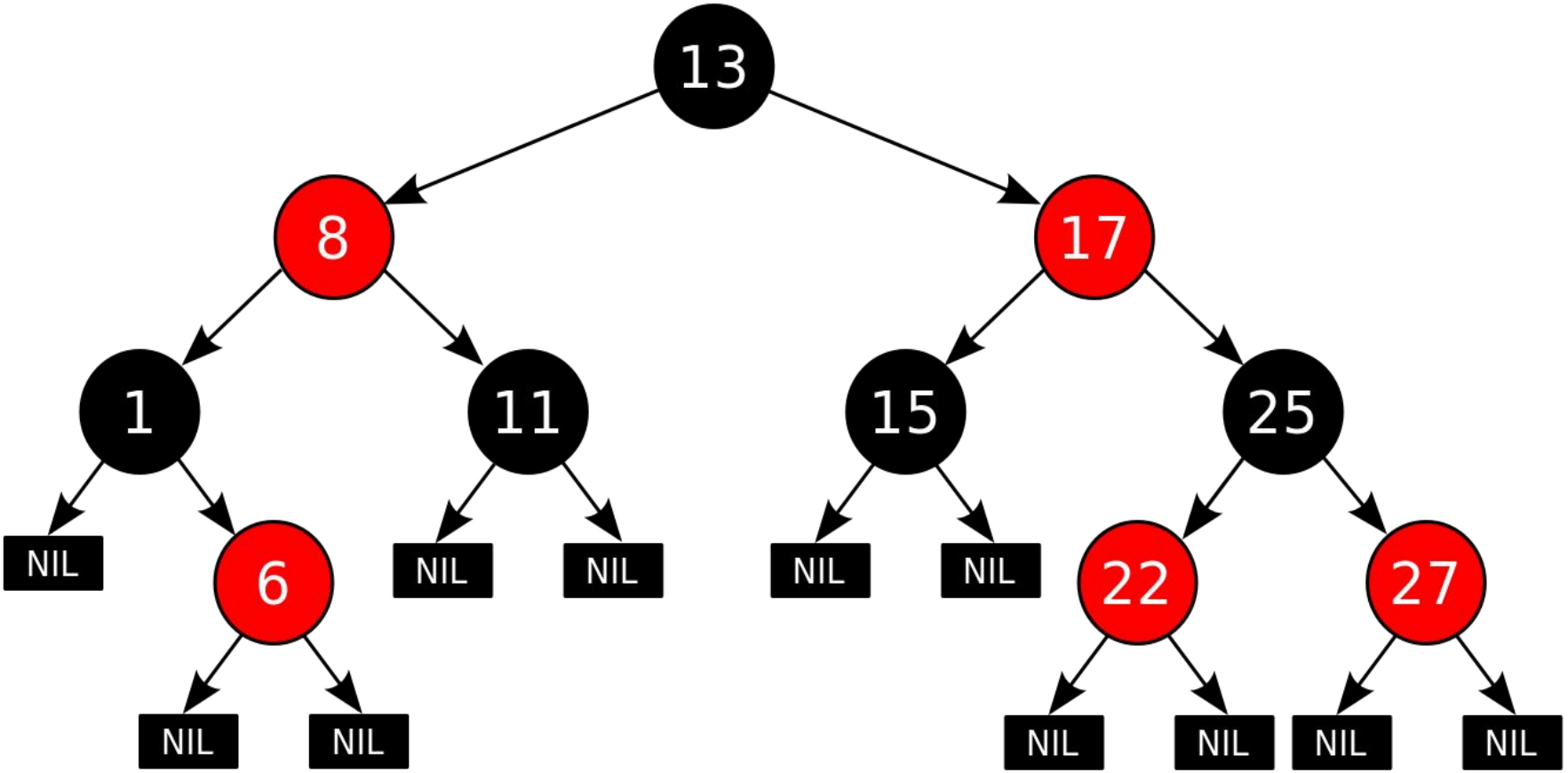
**Красно-чёрное дерево** — это одно из самобалансирующихся двоичных деревьев поиска, гарантирующих логарифмический рост высоты дерева от числа узлов и быстро выполняющее основные операции дерева поиска: добавление, удаление и поиск узла. Сбалансированность достигается за счёт введения дополнительного атрибута узла дерева — «цвета». Этот атрибут может принимать одно из двух возможных значений — «**чёрный**» или «**красный**».

**Красно-чёрные** деревья являются одними из наиболее активно используемых на практике самобалансирующихся деревьев поиска. Популярность **красно-чёрных** деревьев связана с тем, что на них часто достигается подходящий баланс между степенью сбалансированности и сложностью поддержки сбалансированности.

**К красно-чёрным деревьям применяются следующие требования**:

- Узел либо **красный**, либо **чёрный**.
- Корень — **чёрный**.
- Все листья NIL — **чёрные**.
- Оба потомка каждого **красного** узла — **чёрные**.
- Всякий простой путь от данного узла до любого листового узла, являющегося его потомком, содержит одинаковое число **чёрных** узлов.

# КРАСНО-ЧЕРНОЕ ДЕРЕВО



# ХЭШ-ТАБЛИЦЫ

**Хеш-таблица** — это структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу.

**Хэширование** (hashing) — преобразование массива входных данных произвольной длины в (выходную) битовую строку фиксированной длины, выполняемое определённым алгоритмом. Функция, реализующая алгоритм и выполняющая преобразование, называется «**хеш-функцией**». Исходные данные называются входным массивом, «ключом» или «сообщением». Результат преобразования (выходные данные) называется «**хешем**», «**хеш-кодом**», «**хеш-суммой**».

# ХЭШ-ТАБЛИЦЫ

Хеширование применяется в следующих случаях:

- при построении ассоциативных массивов;
- при поиске дубликатов в сериях наборов данных;
- при построении уникальных идентификаторов для наборов данных;
- при вычислении контрольных сумм от данных (сигнала) для последующего обнаружения в них ошибок (возникших случайно или внесённых намеренно), возникающих при хранении и/или передаче данных;
- при сохранении паролей в системах защиты в виде хеш-кода (для восстановления пароля по хеш-коду требуется функция, являющаяся обратной по отношению к использованной хеш-функции);
- при выработке электронной подписи (на практике часто подписывается не само сообщение, а его «хеш-образ»);

# ХЭШ-ТАБЛИЦЫ

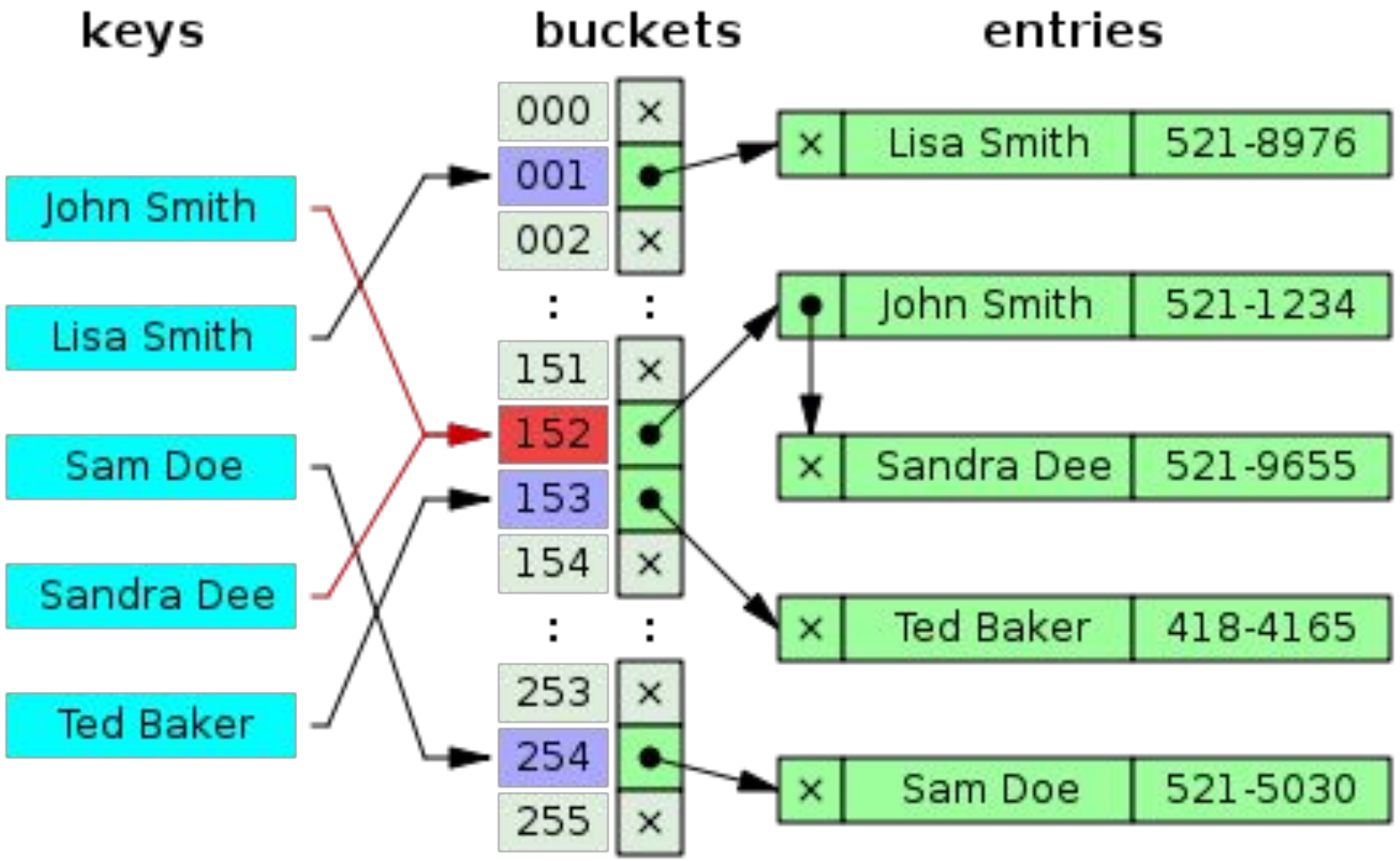
Существуют два основных варианта хеш-таблиц: с **цепочками** и **открытой адресацией**. Хеш-таблица содержит некоторый массив, элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение операции в хеш-таблице начинается с вычисления ***хеш-функции*** от ключа. Затем выполняемая операция (добавление, удаление или поиск) перенаправляется объекту, который хранится в соответствующей ячейке массива.

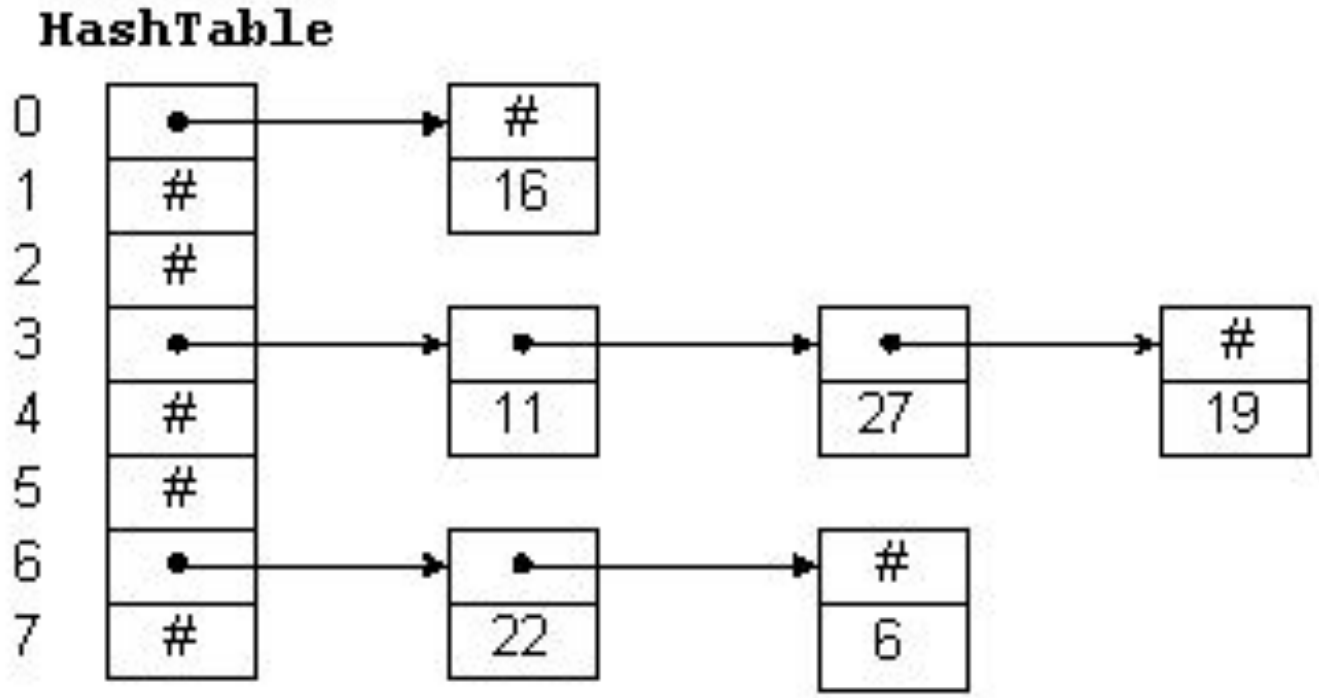
Ситуация, когда для различных ключей получается одно и то же хеш-значение, называется ***коллизией***. Механизм разрешения коллизий — важная составляющая любой хеш-таблицы.

**Важное свойство хеш-таблиц состоит в том, что, при некоторых разумных допущениях, все три операции (поиск, вставка, удаление элементов) в среднем выполняются за время O(1).**

# ХЭШ-ТАБЛИЦЫ


keys
buckets
entries
John Smith
Lisa Smith
Sam Doe
Sandra Dee
Ted Baker
000
001
002
151
152
153
154
253
254
255
Lisa Smith
521-8976
John Smith
521-1234
Sandra Dee
521-9655
Ted Baker
418-4165
Sam Doe
521-5030

# ХЭШ-ТАБЛИЦЫ



# СТРУКТУРЫ ДАННЫХ В ЯЗЫКАХ ПРОГРАММИРОВАНИЯ