Data Structures

Структуры данных

Никлаус Вирт, швейцарский информатик, написал в 1976 году книгу под названием **Алгоритмы + структуры данных = программы**.

Что такое структуры данных?

- **Структура данных** – это контейнер
- который хранит информацию в определенном виде.
- Из-за такой «компоновки» она может быть эффективной в одних операциях и неэффективной в других.
- **Цель разработчика** – выбрать из существующих структур оптимальный для конкретной задачи вариант.

Наиболее часто используемые структуры

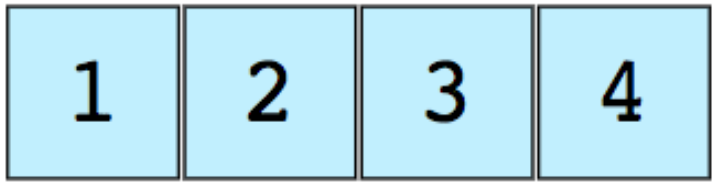
- **Массив (Array)**
- **Стек (Stack)**
- **Очередь (Queue)**
- **Связный список (Linked List)**
- **Дерево (Tree)**
- **Граф (Graph)**
- **Префиксное дерево (Trie)**
- **Хэш-Таблица (Hash Table)**

Массивы

Массивы

- **Массив** – это самая простая и наиболее широко используемая из структур.
- **Стеки** и **очереди** являются производными от массивов.

Массивы



Существует два типа массивов:

- **Одномерные** массивы.
- **Многомерные** массивы (массивы массивов).

Основные операции с массивами

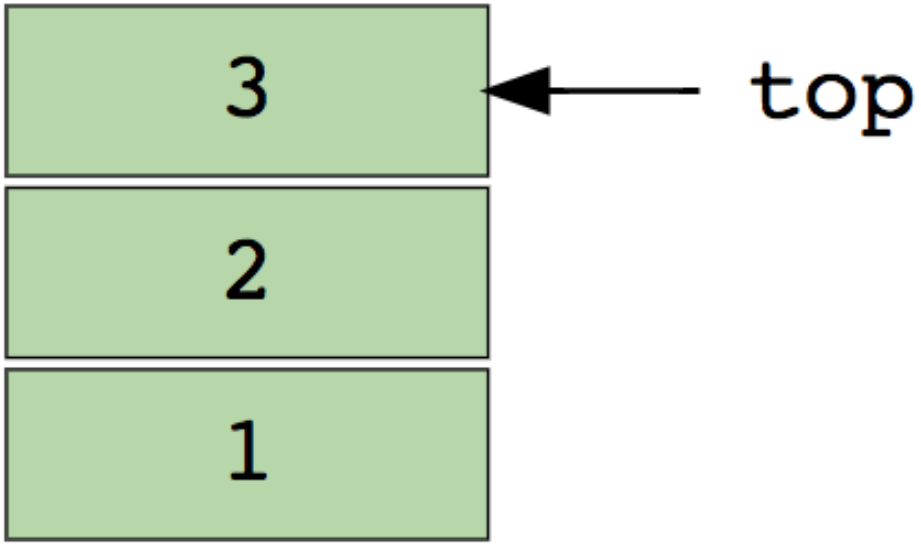
- **Insert** – вставка.
- **Get** – получение элемента.
- **Delete** – удаление.
- **Size** – получение общего количества элементов в массиве.

Стек

Стек

- Пример стека из реальной жизни – куча книг, лежащих друг на друге.
- Чтобы получить книгу, которая находится где-то в середине
- вам нужно удалить все, что лежит сверху.
- Так работает метод **LIFO** (**Last In First Out**, **последним пришел – первым ушел**).

Стек



Основные операции со стеками:

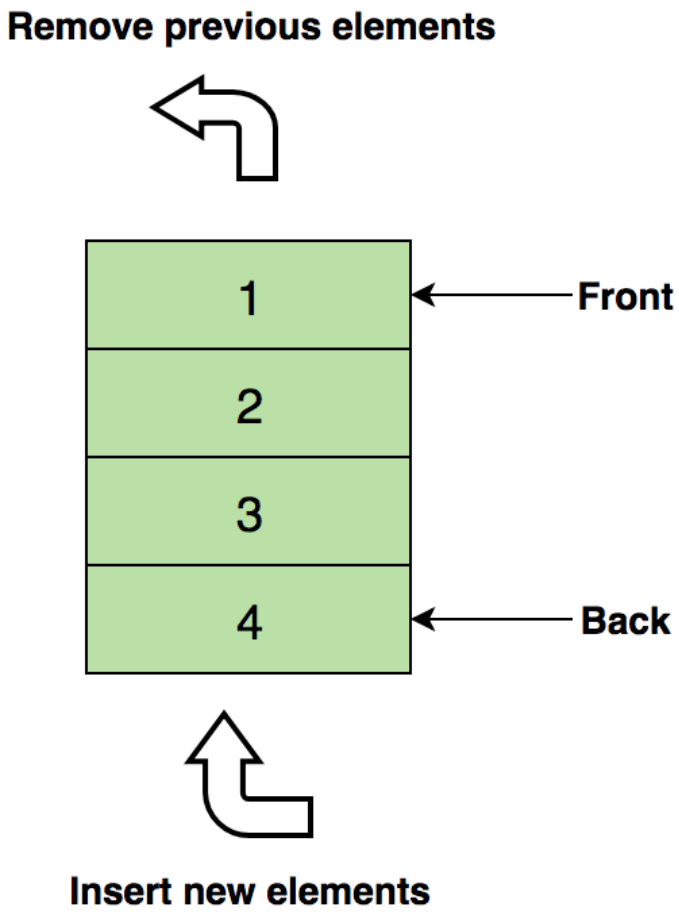
- **Push** – вставка элемента наверх стека.
- **Pop** – получение верхнего элемента и его удаление.
- **isEmpty** – возвращает true, если стек пуст.
- **Top** – получение верхнего элемента без удаления.

Очередь

Очередь

- Как и стек, **очередь** – это линейная структура данных
- которая хранит элементы последовательно.
- Единственное существенное различие заключается в том, что вместо использования метода **LIFO**, очередь реализует метод **FIFO**
- **FIFO** (**First in First Out**, **первым пришел – первым ушел**).

Очередь


Remove previous elements
1
2
3
4
Front
Back
Insert new elements

Основные операции с очередями:

- **Enqueue** – вставка в конец.
- **Dequeue** – удаление из начала.
- **isEmpty** – возвращает true, если очередь пуста.
- **Top** – получение первого элемента.

Связный список

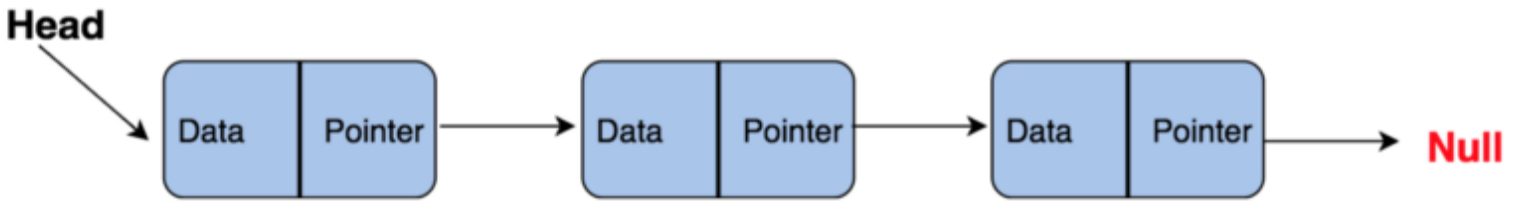
Связный список

- Линейная структура данных
- которая на первый взгляд похожа на массив
- но отличается:
  - распределением памяти
  - внутренней организацией
  - способом выполнения основных операций вставки и удаления

Связный список

- **Связный список** – это сеть узлов, каждый из которых содержит данные и указатель на следующий узел в цепочке.
- Также есть указатель на первый элемент – `head`.
- Если список пуст, то он указывает на `null`.
- Связные списки используются для реализации файловых систем, хэш-таблиц и списков смежности.

Связный список



Типы связных списков:

- **Однонаправленный**
- **Двунаправленный**

Основные операции со связными списками

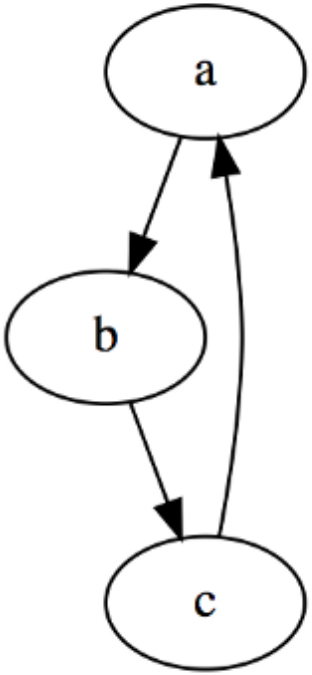
- **InsertAtEnd** – вставка в конец.
- **InsertAtHead** – вставка в начало.
- **Delete** – удаление указанного элемента.
- **DeleteAtHead** – удаление первого элемента.
- **Search** – получение указанного элемента.
- **isEmpty** – возвращает true, если связный список пуст.

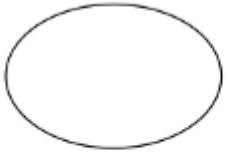
Графы

Графы

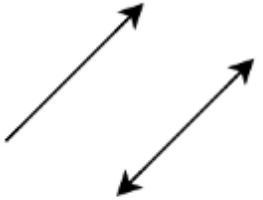
- **Граф** представляет собой набор узлов, соединенных друг с другом в виде сети.
- **Узлы** также называются **вершинами**.
- Пара (x, y) называется **ребром**, которое указывает, что вершина x соединена с вершиной y.
- Ребро может содержать вес/стоимость, показывая, сколько затрат требуется, чтобы пройти от x до y.

Графы





Vertex



Edge

Типы графов

- **Неориентированный**
- **Ориентированный**

В языке программирования графы могут быть представлены в двух **формах**:

- **Матрица смежности**
- **Список смежности**

Общие алгоритмы обхода графов:

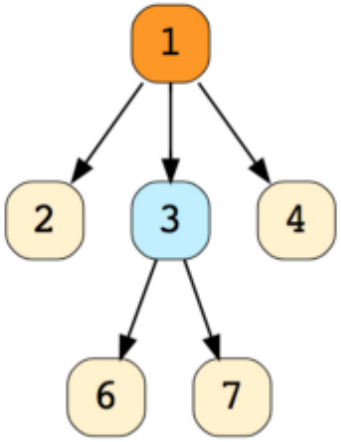
- **В ширину**
- **В глубину**

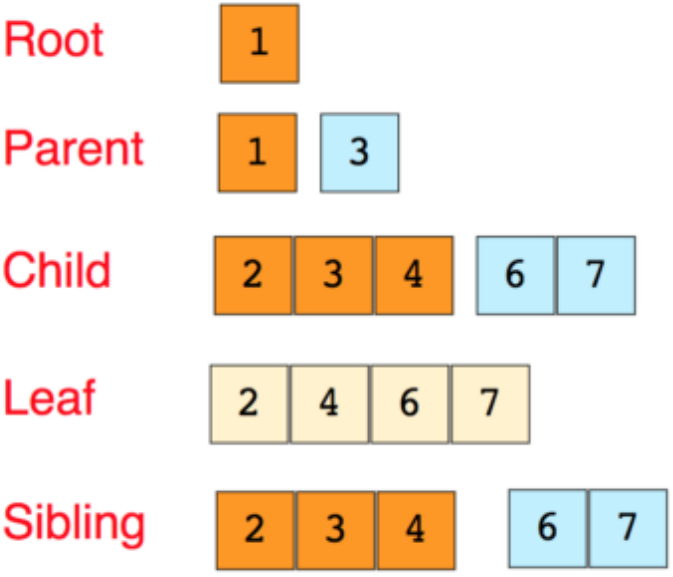
Деревья

Деревья

- **Дерево** – это иерархическая структура данных, состоящая из **вершин** (**узлов**) и **ребер**, соединяющих их.
- Они похожи на графы, но есть одно важное отличие: в дереве не может быть цикла.

Деревья





Типы деревьев

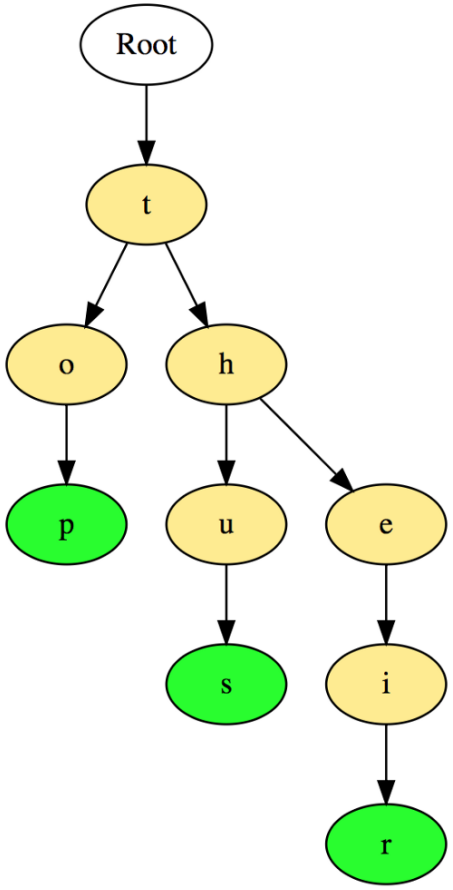
- N-арное дерево;
- сбалансированное дерево;
- бинарное дерево;
- бинарное дерево поиска;
- дерево AVL;
- красно-чёрное дерево;
- 2-3 дерево.

Префиксное дерево

Префиксное дерево

- **Префиксные деревья** (**tries**) – древовидные структуры данных, эффективные для решения задач со строками.
- Они обеспечивают быстрый поиск и используются преимущественно для поиска слов в словаре, автодополнения в поисковых системах и даже для IP-маршрутизации.

Префиксное дерево



Хеш-Таблица

Хеш-Таблица

- **Хеширование** – это процесс, используемый для уникальной идентификации объектов и хранения каждого из них в некотором предварительно вычисленном уникальном индексе – **ключе**.
- Итак, объект хранится в виде пары ключ-значение, а коллекция таких элементов называется **словарем**.

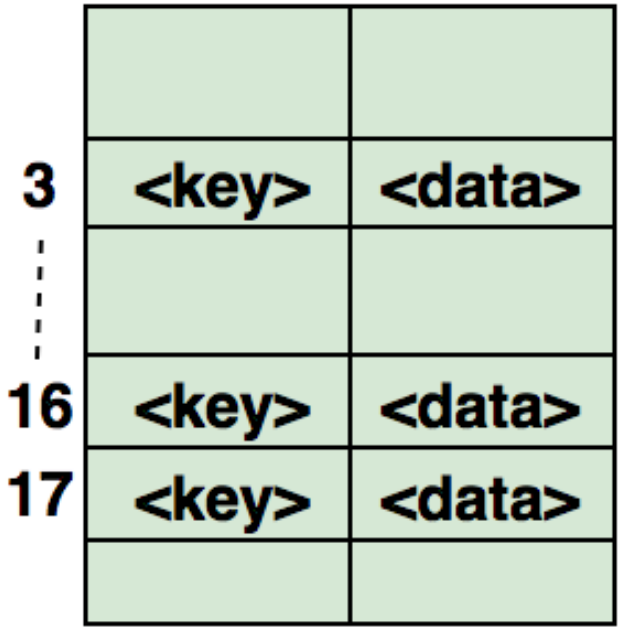
Хеш-Таблица

- Каждый объект можно найти с помощью его ключа.
- Существует несколько структур, основанных на хешировании, но наиболее часто используется **хеш-таблица**, которая обычно реализуется с помощью массивов.

Хеш-Таблица

Производительность структуры зависит от трех факторов:

- функция хеширования
- размер хеш-таблицы
- метод обработки коллизий


3
16
17
<key>
<data>
<key>
<data>
<key>
<data>