Тема рассчитана на 2-3 занятия, проходить лучше медленно и с подробными объяснениями.

Показать картинки, рассказать об однонаправленных, двунаправленных и циклических списках

Связные списки нужны там, где с одной стороны нужны листы/наборы данных. А с другой — быстрые вставки в середину. Примеров не много, но вот два:  
1) Нам периодически приходят обновления о том, что объект из списка пропал. И происходит это часто, а объектов много. Например нам сообщают, что машина продана.  
2) Удобно составлять цепи событий, куда можно вставлять промежуточные события.

Связные списки используются в хэш-таблицах.

Вопрос студентам – применим ли бинарный поиск к связным спискам? – ответ: нет, т.к. это дольше, чем просто пройтись по списку.

Преимущества связных списков

* эффективное (за константное время) добавление и удаление элементов
* размер ограничен только объёмом памяти [компьютера](https://ru.wikipedia.org/wiki/%D0%AD%D0%92%D0%9C) и разрядностью указателей
* динамическое добавление и удаление элементов

Недостатки связных списков вытекают из их главного свойства — последовательного доступа к данным:

* сложность прямого доступа к элементу, а именно определения физического адреса по его [индексу](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81) (порядковому номеру) в списке
* на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в [массивах](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), например, указатели не нужны)
* многие операции со списками медленнее, чем с массивами, так как к произвольному элементу списка можно обратиться, только пройдя все предшествующие ему элементы
* соседние элементы списка могут быть распределены в памяти нелокально, что снизит эффективность [кэширования](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88) данных в процессоре
* над связными списками, по сравнению с массивами, гораздо труднее (хоть и возможно) производить параллельные векторные операции, такие, как вычисление суммы: накладные расходы на перебор элементов снижают эффективность распараллеливания

Практика - прога 1 - однонаправленный связный список (сделать добавление элемента, расчет длины, удаление элемента)

В практике сделать самому пару функций, потом пусть они делают – проверять их.

Потом отделить список в отдельный файл – сделать класс через прототипы (Как в "Еще программы/Все проги 2018/Single\_list.js")

Потом сделать двунаправленный список (те же функции)

Потом циклический (те же функции)

Задание со звездочкой - сделать разворот списка без создания его копии

В файле "Задания" есть доп задания для успевающих

В ДЗ задавать доделывать функции и т.п.