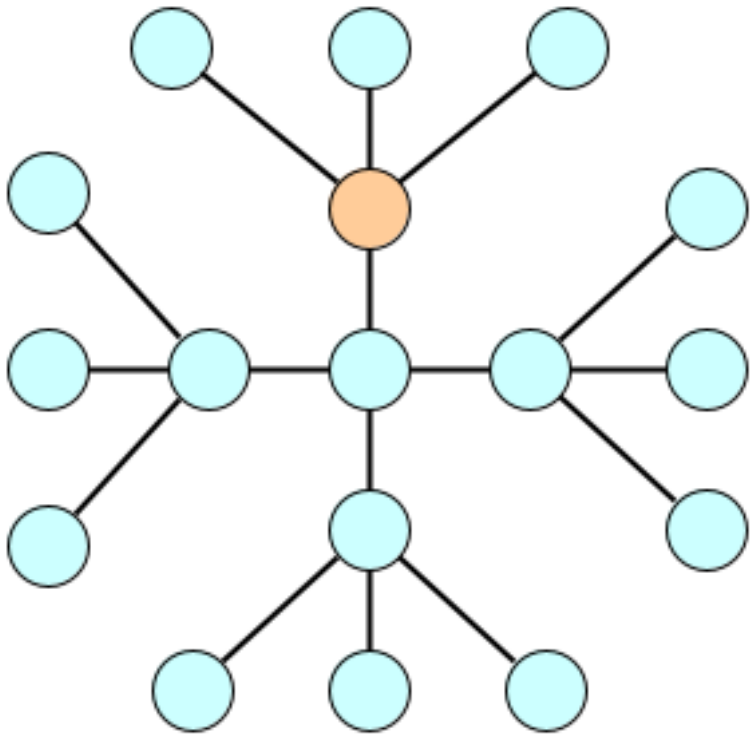
БИНАРНЫЕ ДЕРЕВЬЯ

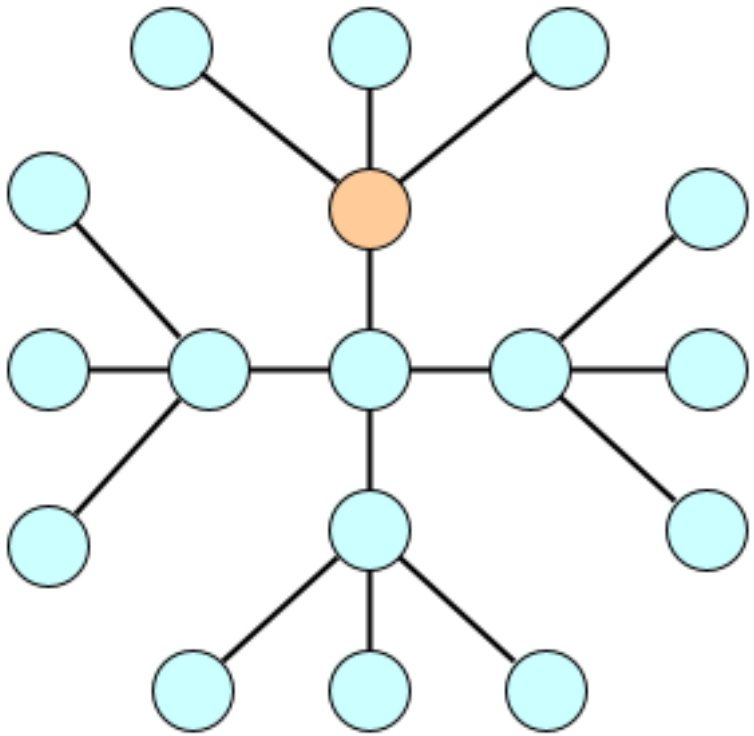
Их ветви надо мной уходят в темноту

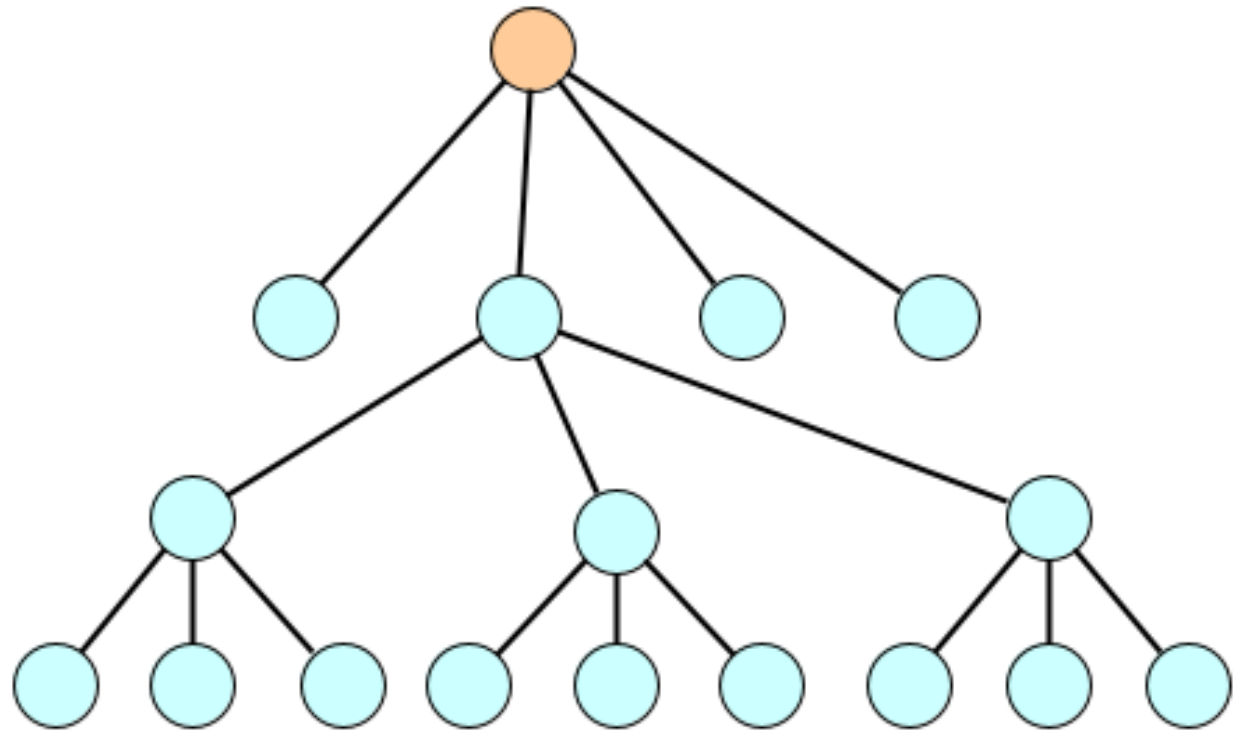
К. Владимиров, Intel, 2020
mail-to: konstantin.vladimirov@gmail.com

Что перед вами?

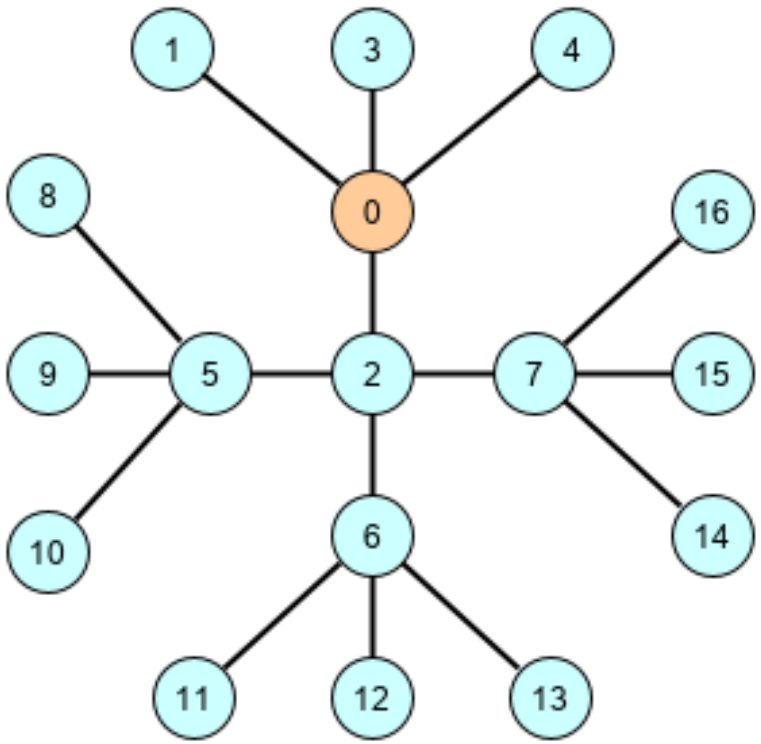


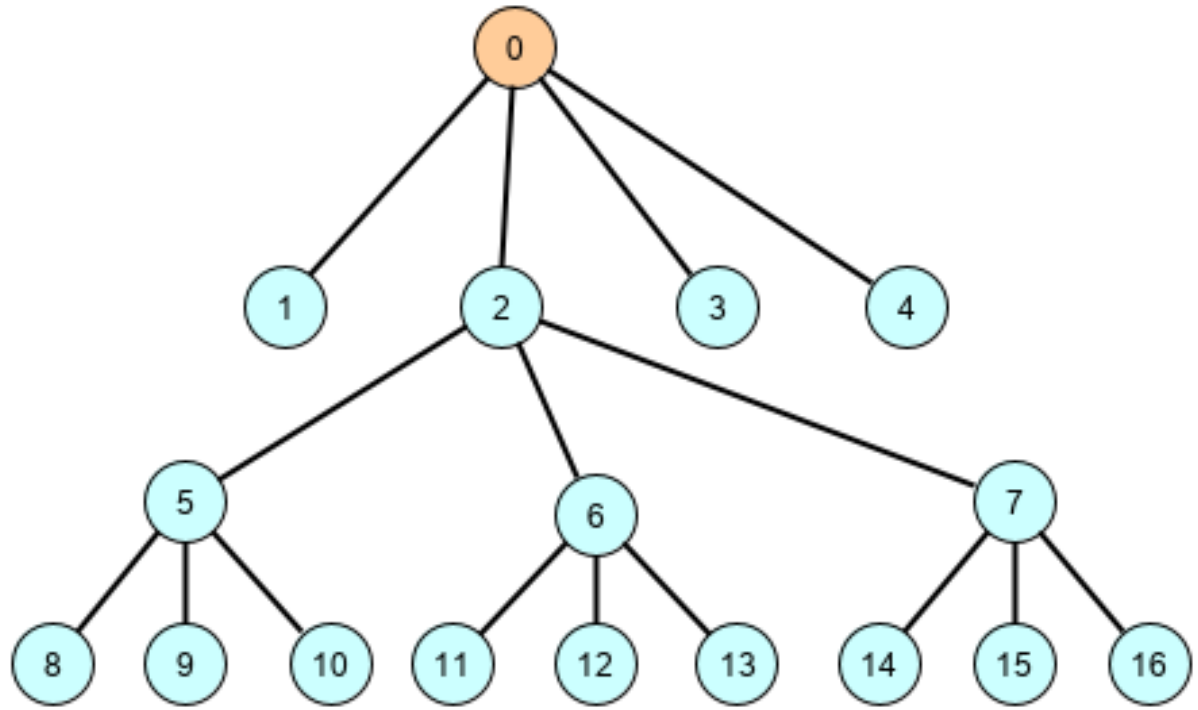
Что перед вами?





Что перед вами?





Как представить дерево на языке C

- Структура данных содержит элемент который ссылается на себя

```
struct Node_t {
  // массив указателей на детей (а почему 10?)
  struct Node_t* nodes[10];

  // количество детей
  int nchilds;

  // данные в узле
  int data;
};
```

- Лучшие варианты?

Как представить дерево на языке C

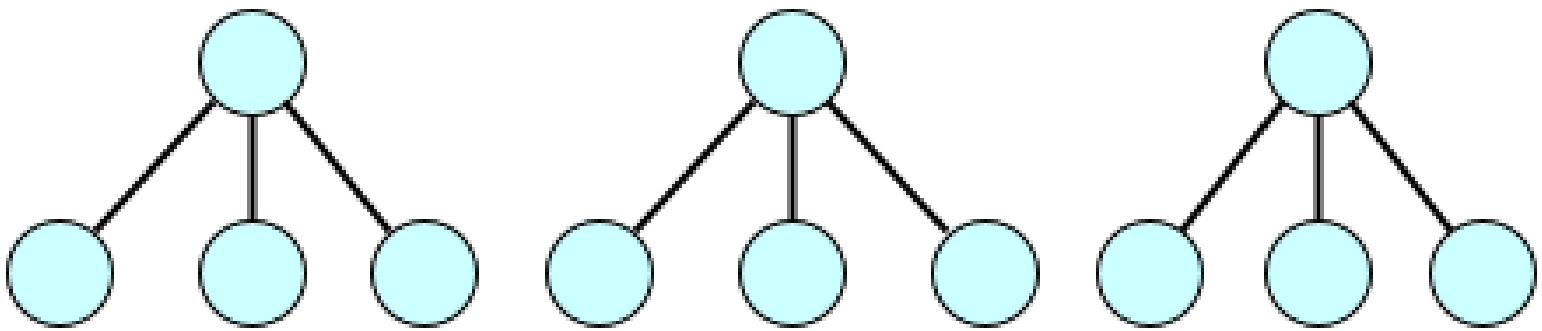
- Структура данных содержит элемент который ссылается на себя

```
struct Node_t {
  // массив указателей на детей (выделяем динамически)
  struct Node_t** nodes;

  // количество детей
  int nchilds;

  // данные в узле
  int data;
};
```

- Как-то сложно…

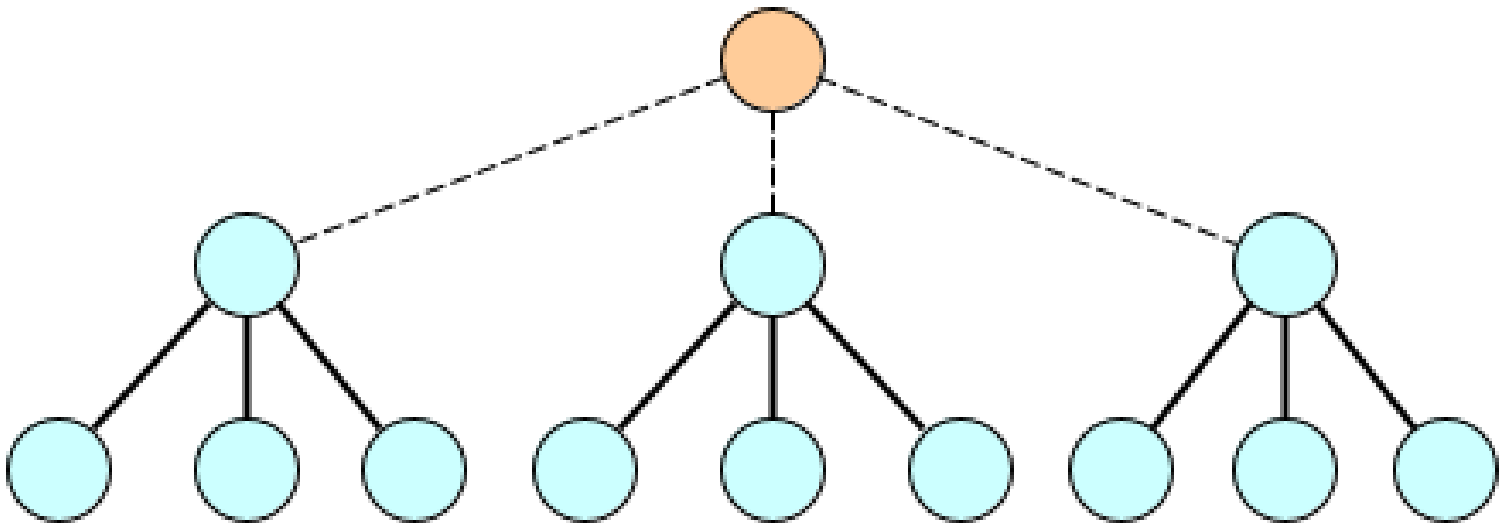
Немного о лесах

- Лес это одно или несколько деревьев. Как представить лес в программе на C?



Немного о лесах

- Лес это одно или несколько деревьев. Как представить лес в программе на C?
- Оказывается лес это дерево без вершины

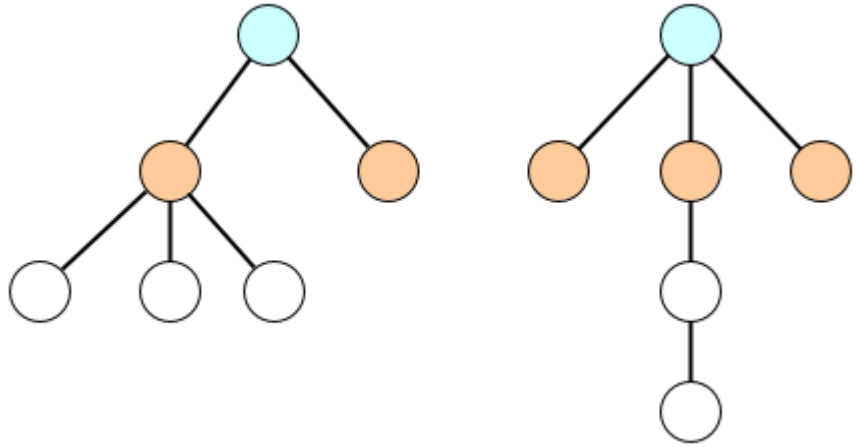


Скобочное выражение это лес

- Рассмотрим правильную расстановку скобок

((()()())())(()(()))())

- Довольно очевидно, что это лес
- Но такое чувство что это не самое удобное представление для работы

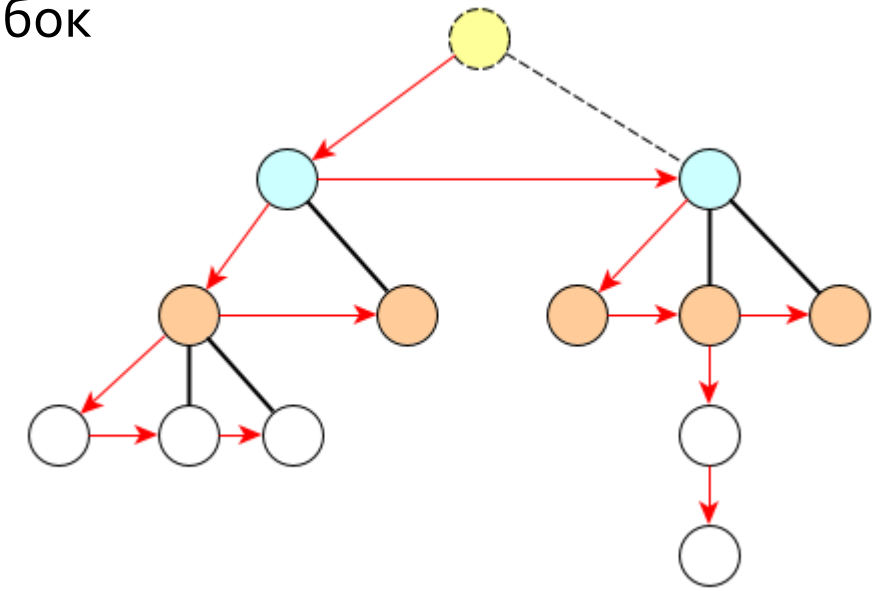


Скобочное выражение это лес

- Рассмотрим правильную расстановку скобок

((()()())())(()(()))())

- Довольно очевидно, что это лес
- Но такое чувство что это не самое удобное представление для работы
- Проведём стрелки к первому брату и к первому потомку
- И внезапно мы видим…

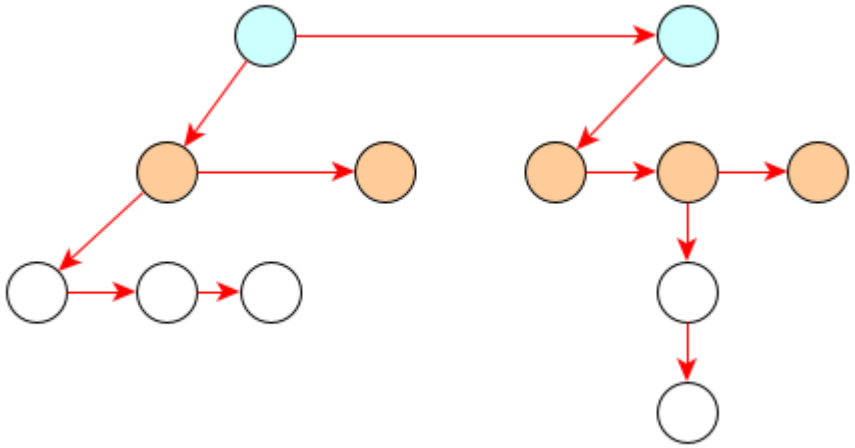


Скобочное выражение это лес

- Рассмотрим правильную расстановку скобок

((()()())())(()(()))())

- Довольно очевидно, что это лес
- Но такое чувство что это не самое удобное представление для работы
- Проведём стрелки к первому брату и к первому потомку

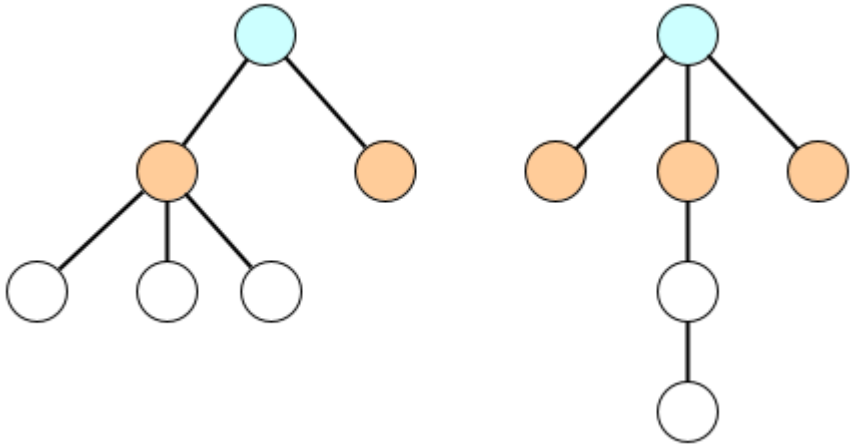


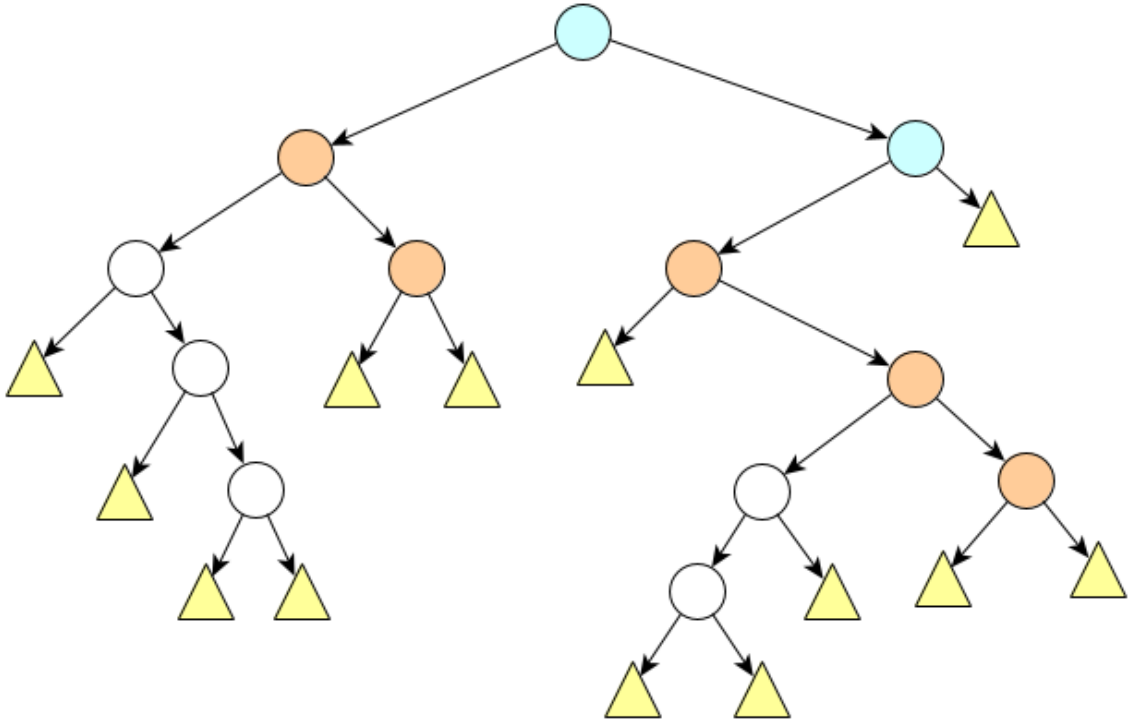
- И внезапно мы видим что у нас получилась древовидная структура с двумя связями на каждый узел
- Итак любой лес (а также любая перестановка, сочетание и разбиение) это…

Лес это бинарное дерево

- Внезапно это бинарное дерево

((()()())())(()(()))())



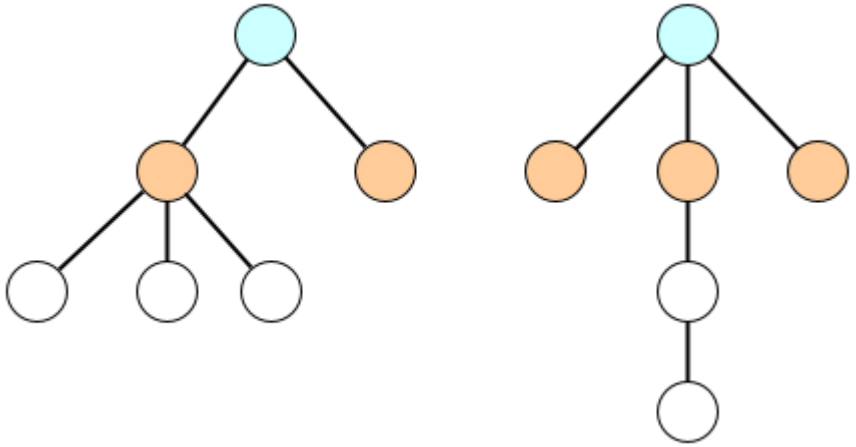


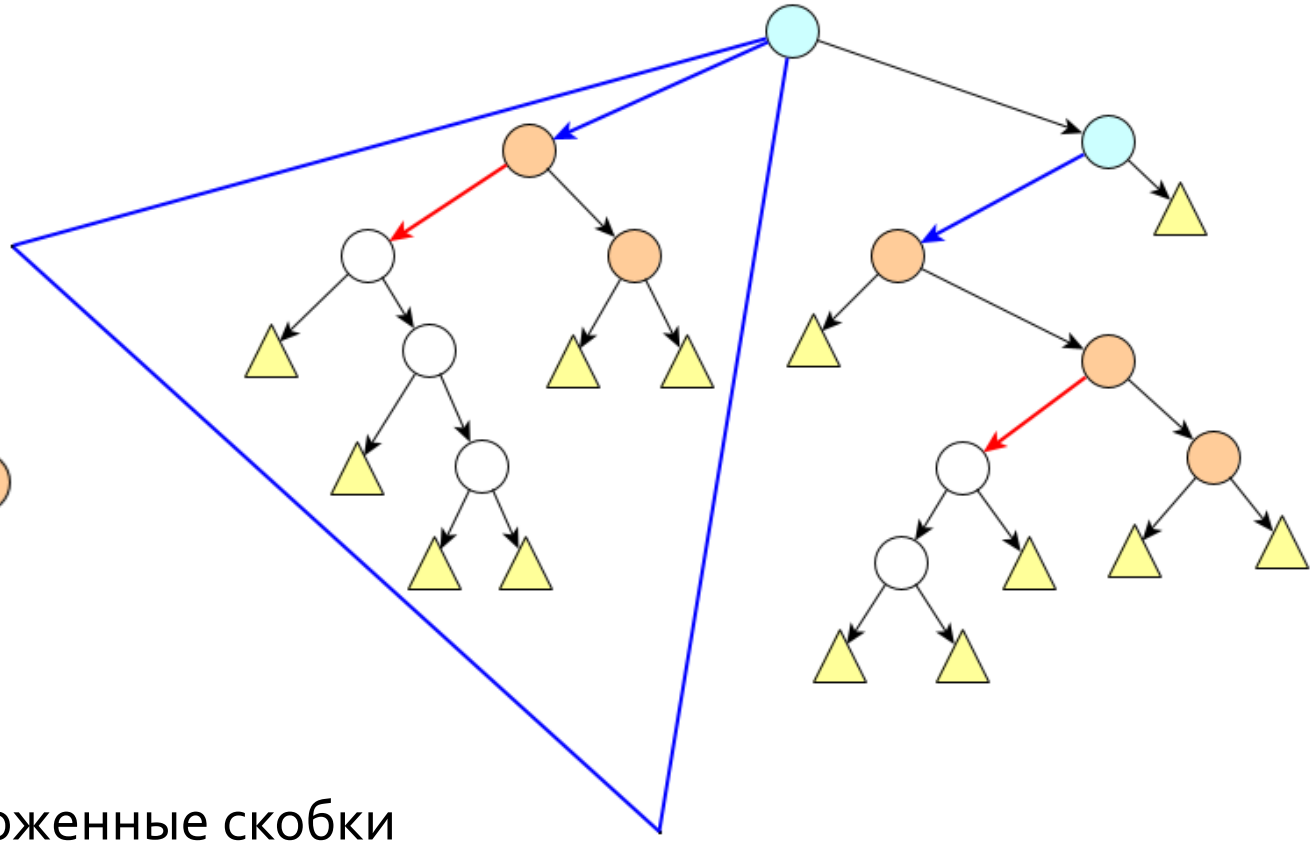
- Кстати, а вы увидели в бинарном дереве последовательность скобок?

Лес это бинарное дерево

- Внезапно это бинарное дерево

((()()())())(()(()))())





- Если есть левый потомок, это вложенные скобки
- Если есть правый потомок это скобки, стоящие рядом

Бинарные деревья в языке C

- Представление бинарного дерева приятно просто

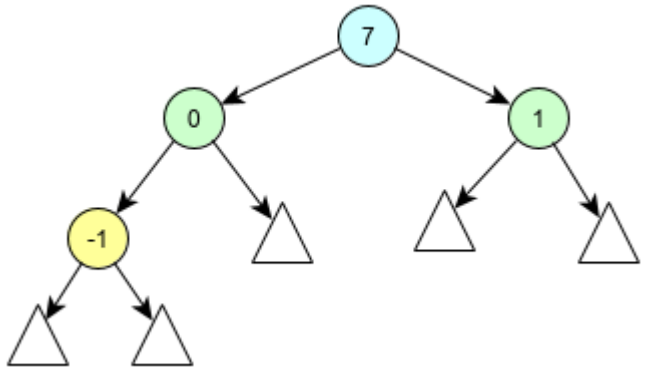
```
struct BNode_t {
  // левый и правый потомки (возможно NULL)
  struct BNode_t* left;
  struct BNode_t* right;

  // данные в узле
  int data;
};
```

- Но как сохранить бинарное дерево в строку или в файл?
- Конечно можно записать нечто вроде 1(2(4()5()6())3())

Упражнение

- Сделайте набор функций чтобы в программе создать простое дерево

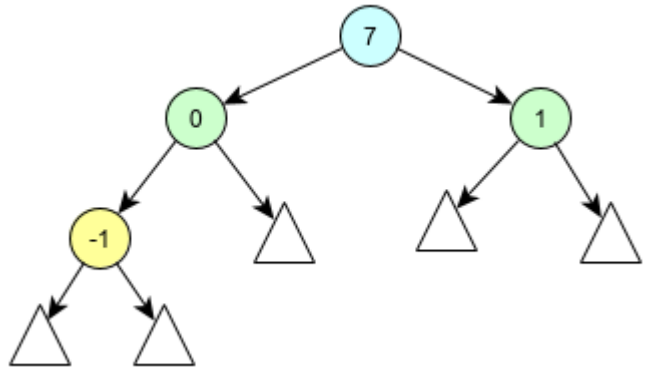
```
struct BNode_t *n1, *n2, *n3, *top;
n2 = create_node(-1, NULL, NULL);
n1 = create_node(0, n2, NULL);
n3 = create_node(1, NULL, NULL);
top = create_node(7, n1, n3);
free_tree(top);
```



Упражнение

- Распечатайте простое дерево иерархически

```
7
..0
....-1
..1
```



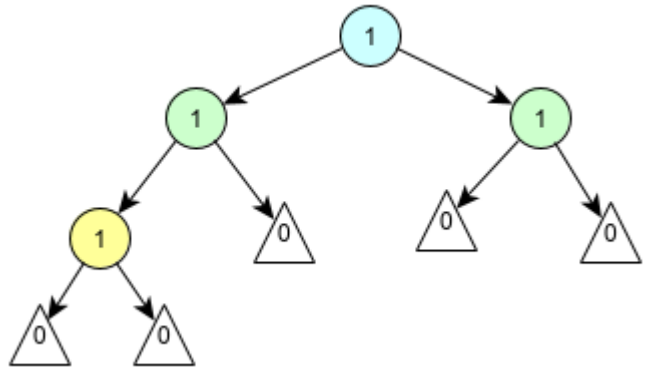
- Распечатайте дерево как список рёбер

```
7 0 7 1 0 -1
```

- Распечатайте бинарное дерево как скобочное выражение: `7(0(-1()))1()`
- Хорошая ли идея хранить деревья в файле любым из этих способов?

Бинарные деревья и 0-1 представление

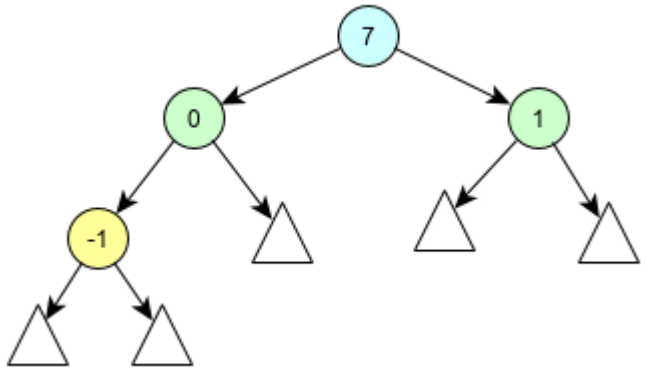
- Любое бинарное дерево соответствует бинарной строке из нулей и единиц
- 1 1 1 0 0 0 1 0 0
- Очевидно нулей должно быть на один больше, так что последний можно сэкономить
- Приведите пример строки с правильным числом нулей и единиц, которой не соответствует ни одно дерево?
- Можем ли мы сохранить некие данные в листьях?



Бинарные деревья и 0-1 представление

- Основная идея: два раздельных потока для структуры и для ключей

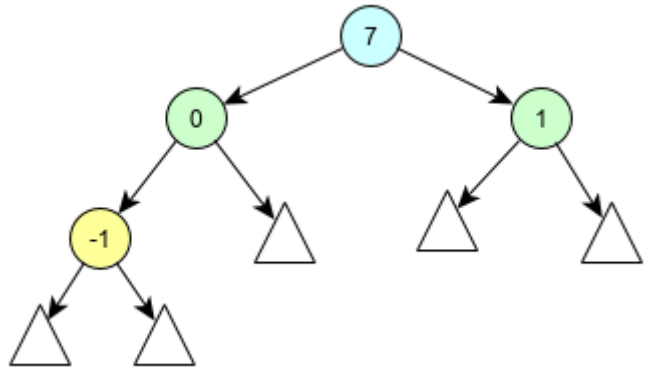
```
1 1 1 0 0 0 1 0
7 0 -1 1
```



- Последний ноль не указан но он есть
- Мы можем закодировать дерево из 32-х узлов в 64-битном числе
- Если мы будем хранить в дереве из 32-х узлов целые числа, мы потратим 128 байт на ключи и только 8 байт на структуру дерева

Problem CO -- на нечётных уровнях

- Считайте `0-1` бинарное дерево, заданное только своей топологией из файла
- Корень находится на нулевом уровне
- Сколько узлов в нём находится на нечётных уровнях?
- Например на входе: `1 1 1 0 0 0 1 0`
- Оба зелёных узла на первом уровне
- На выходе: 2



Problem GT -- генерация и повороты

- Сгенерируйте `N` единиц и `N + 1` нулей
- Случайно их перемешайте посредством Fisher Yates shuffle
- Вы получили равномерно распределённое число из `2N + 1` бит
- Утверждается что всегда можно найти единственный циклический поворот, такой, чтобы это число стала возможным деревом
- Пример: `0 0 1 0 0 1 1 1 0 -> 1 1 1 0 0 0 1 0 0`
- Найдите такой поворот и распечатайте получившееся дерево (увы такие деревья не будут в точности равномерно распределены)

Problem VT -- посещение деревьев

Погуглите алгоритмы прежде чем решать эти задачи, но, пожалуйста, не гуглите реализацию, потренируйтесь самостоятельно

- Посетите все различные бинарные деревья с N внутренними узлами, не посещая ни одно из них дважды. Проверьте себя:
  - для трёх узлов ваша программа должна построить пять различных бинарных деревьев, см. картинку слева
  - для 11 узлов ваша программа должна построить 58786 различных бинарных деревьев
- Сгенерируйте равновероятное случайное дерево с N внутренними узлами