Лекция 8

Динамические структуры данных

План лекции

Структуры данных
- Динамические массивы
- Стек, дека и очередь
- Связные списки
- Бинарное дерево поиска

Вопросы к зачету

Массивы

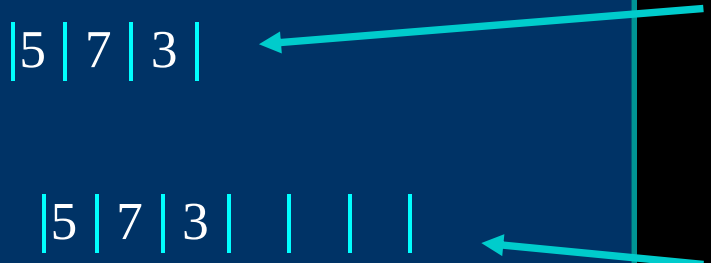
Контейнеры — структуры, предназначенные для хранения набора данных

Си: статический контейнер – массив фиксированный размерности

```
int a[100];
int size = 100;
```

Динамически выделяемая память

```
int *ap;
int size = 3;
…
ap = (int*) malloc(sizeof(int)*size)
if (ap==NULL) …
size*=2;
ap = realloc(ap, sizeof(int)*size)
```



Динамические массивы

Типовые операции

1) Прочитать записать i-ый элемент: ap[i] = 19;
   - константное время

2) Вставить элемент

Новый массив

3) Удалить элемент

4) Поиск элемента по его значению

Линейно зависит от размера массива

Вставка-удаление элементов в конце массива

```
struct array
{
        int *ap;
        int size;
        int tot;
}
```

Стек:

push() – добавить элемент

pop() – извлечь элемент

Принцип L I F O

Двусторонняя вставка-удаление (очередь) - дека

```
struct deque
{
        int *ap;
        int first; // индекс первого реального элемента
        int last; // индекс элемента за последним
}
```

Поддерживаются операции (константное время)

- push_start()
- pop_start()
- push_end()
- pop_end()

Направленная очередь

Поддерживаются операции

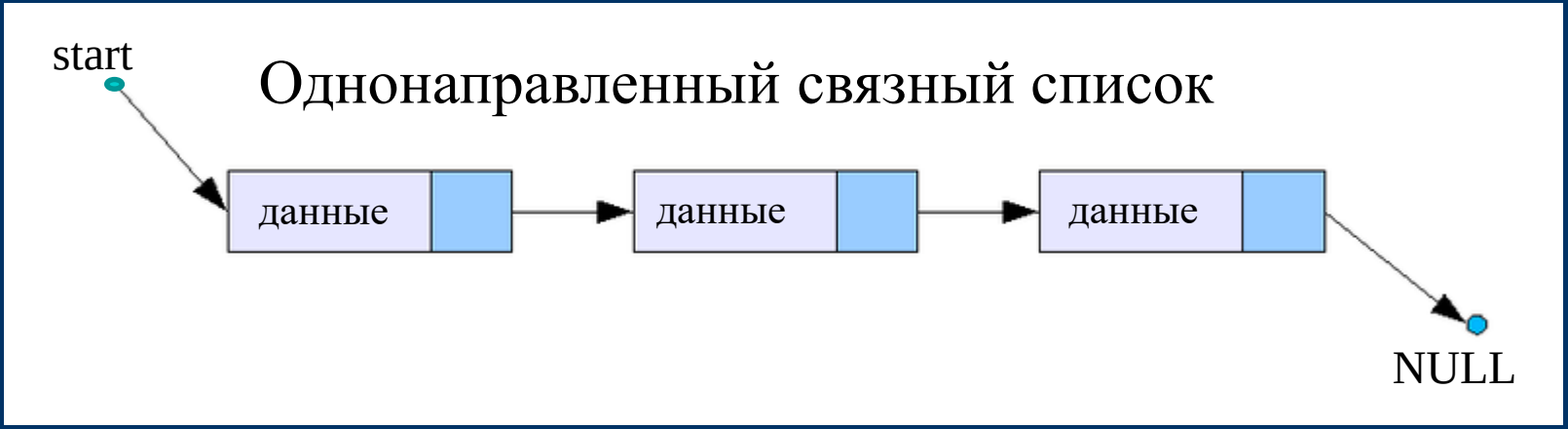
pop_start()
push_end()

Принцип F I F O

Связные списки

(вставка в произвольное место – константное время)

Свя́зный спи́сок — структура данных, состоящая из узлов (элементов), каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка



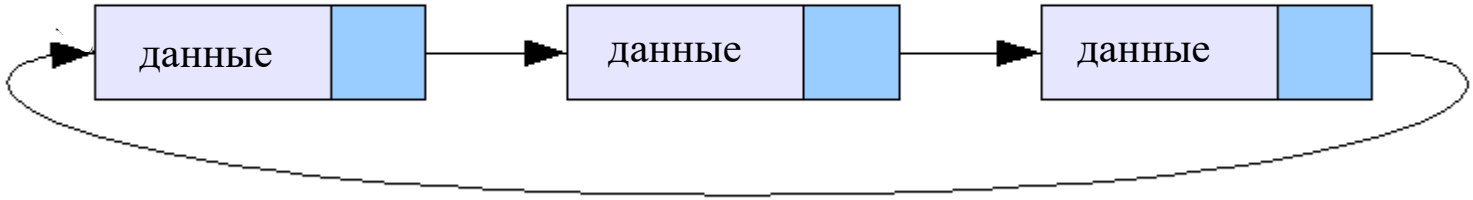
Пример связного списка

a[100] next

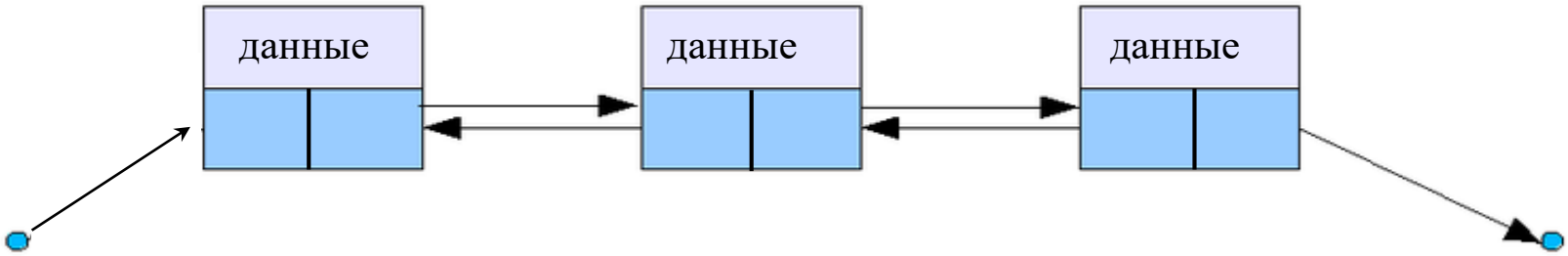

```
                    struct list{ int a[100];          struct listHead{
                            struct list* next;};          struct list* start;
                                                          unsigned size;} ls;
void main(){
  ls.start = (struct list*)malloc(sizeof (struct list));
  ls.start->next = NULL;
  ls.size = 1;
};
```

Связные списки

Кольцевой однонаправленный связный список

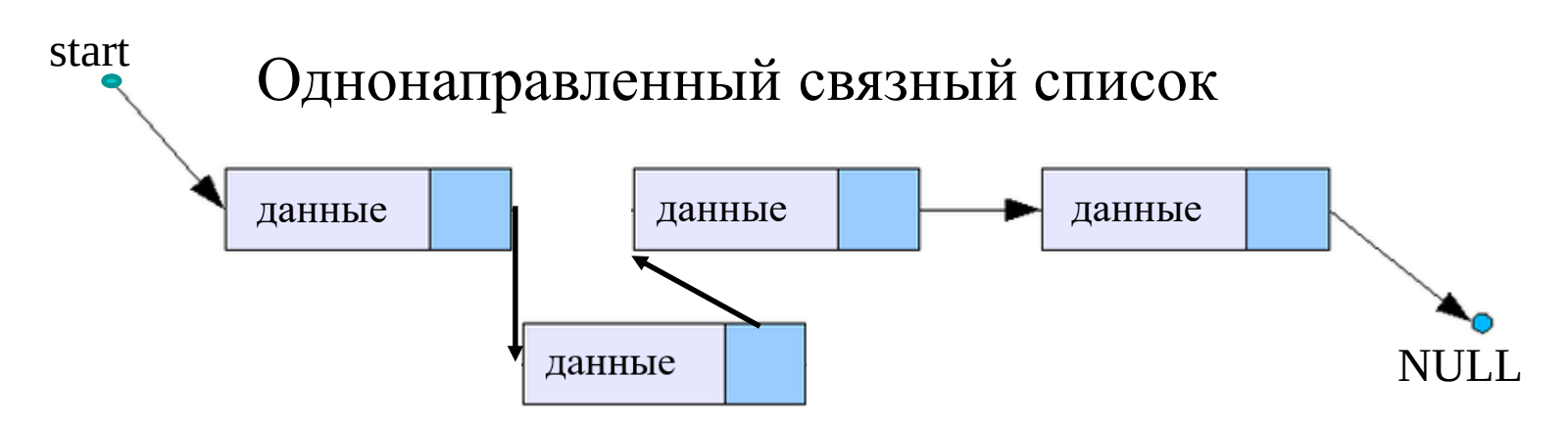


Двунаправленный связный список



Вставка элемента в список

(вставка в произвольное место – константное время)



Операция нахождение элемента списка по его номеру (как и по значению) – линейная!

Бинарное дерево поиска

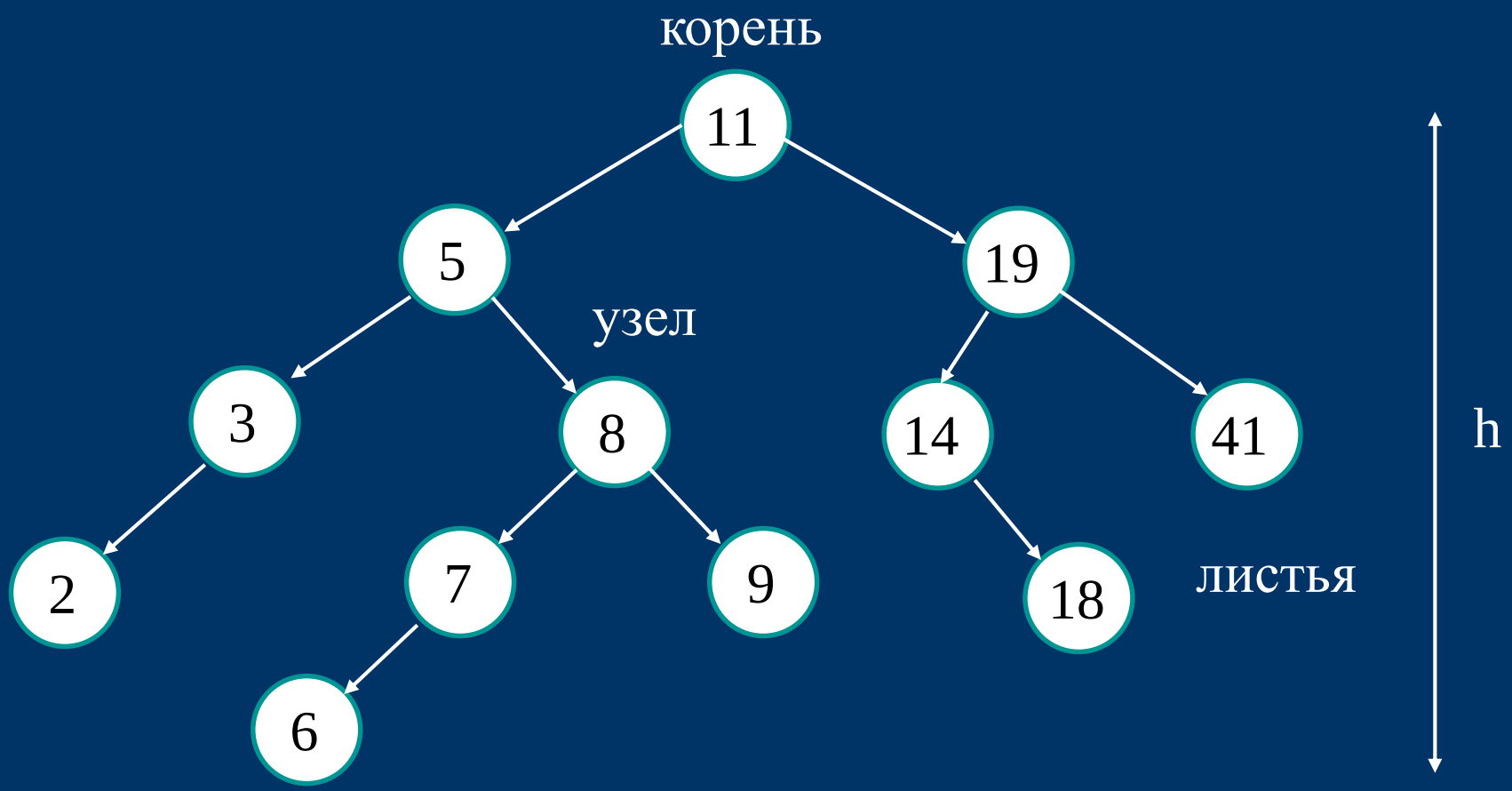
Основная операция – поиск со вставкой

Списки (даже отсортированные) не обеспечат логарифмическое время поиска

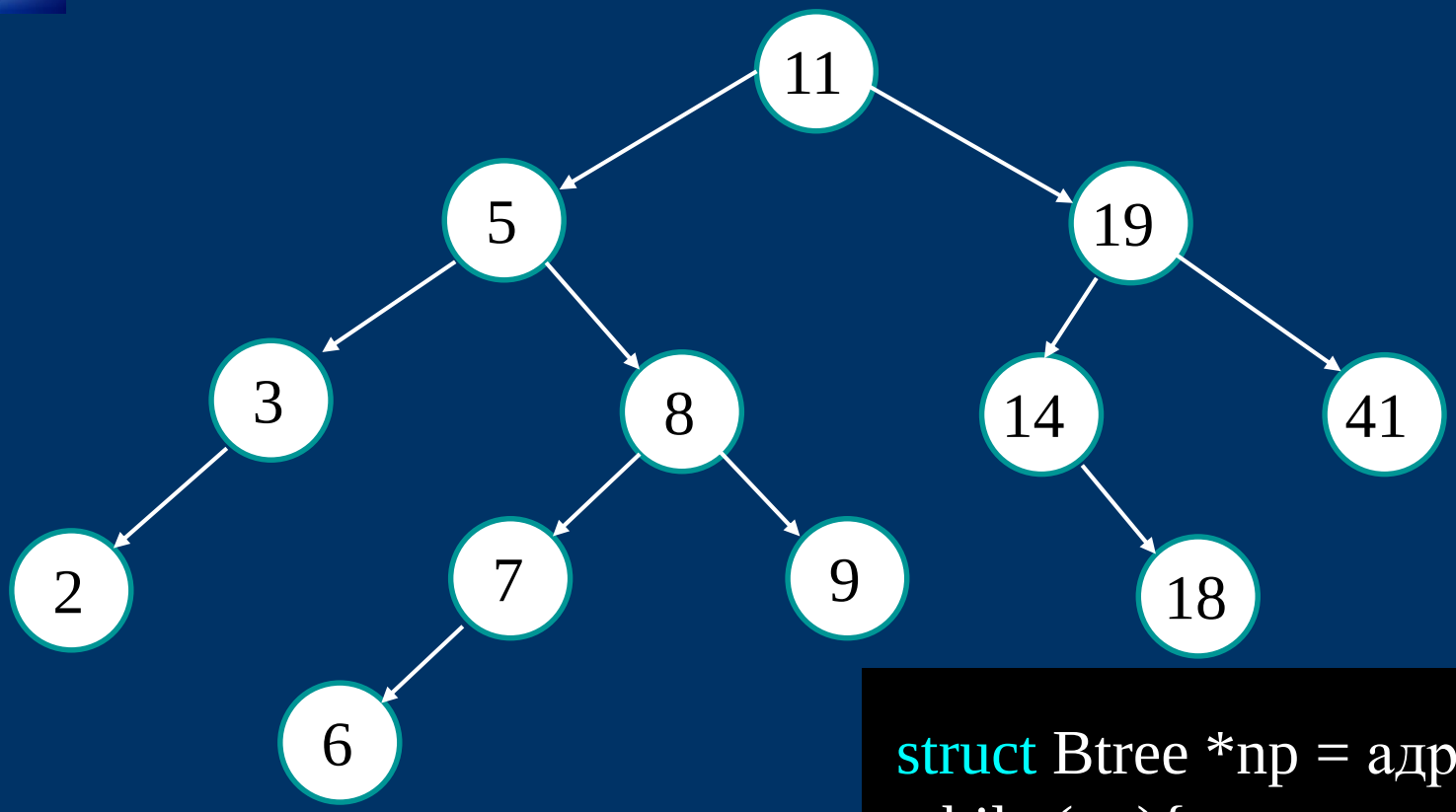
Бинарное дерево – каждый элемент содержит два указателя – на левое и правое поддерево. Узлы дерева поддерживают в определенном порядке

```
struct Btree{
   int a;
   struct Btree *left;
   struct Btree *right;
};
```

Пример бинарного дерева поиска

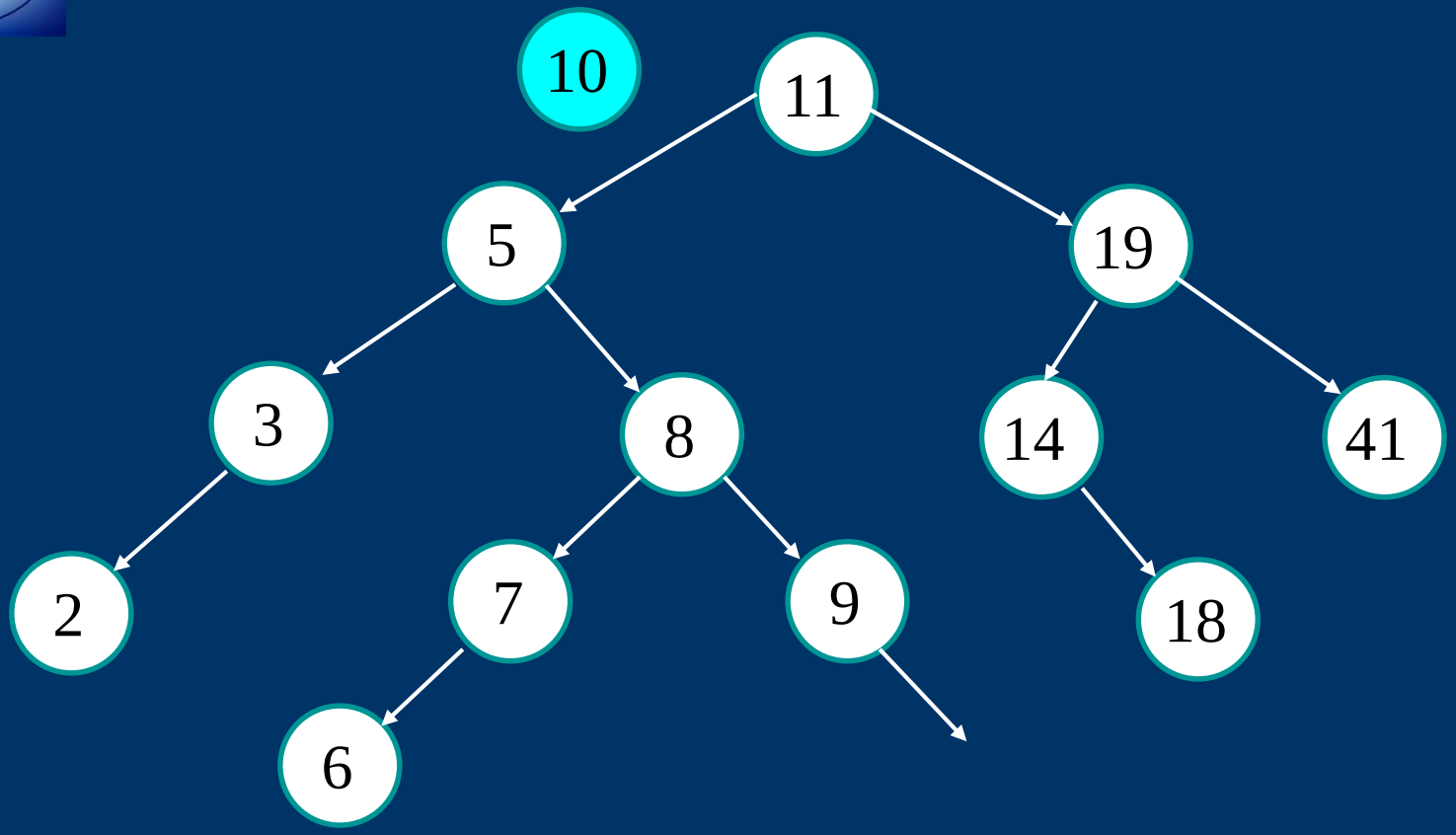


Поиск значения x в бинарном дереве

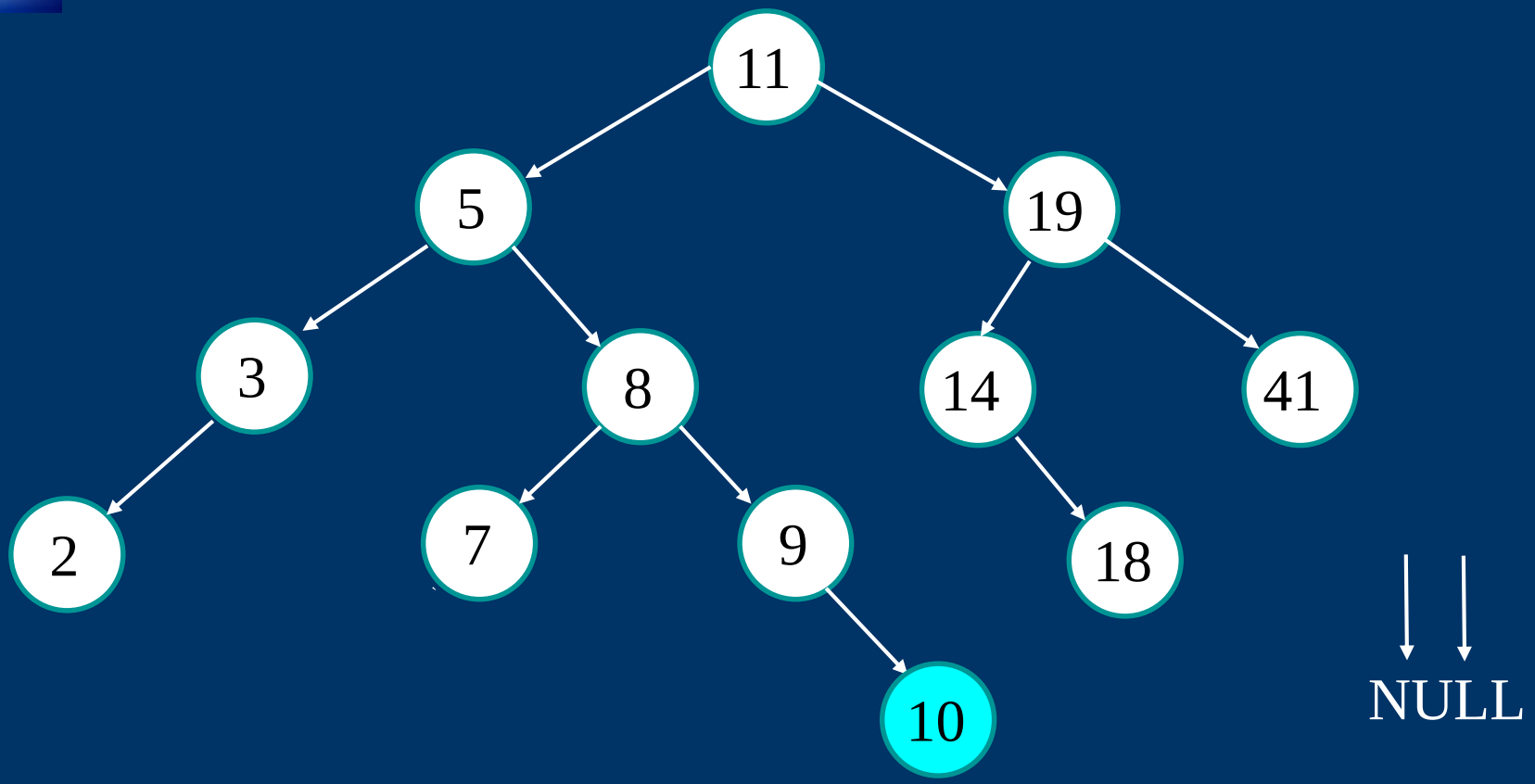



```
struct Btree *np = адрес корня;
while (np){
  if (x== np->a) break; //нашли
  else if (x  < np->a) np=np->left;
  else np=np->right; }
```

Вставка элемента в бинарное дереве

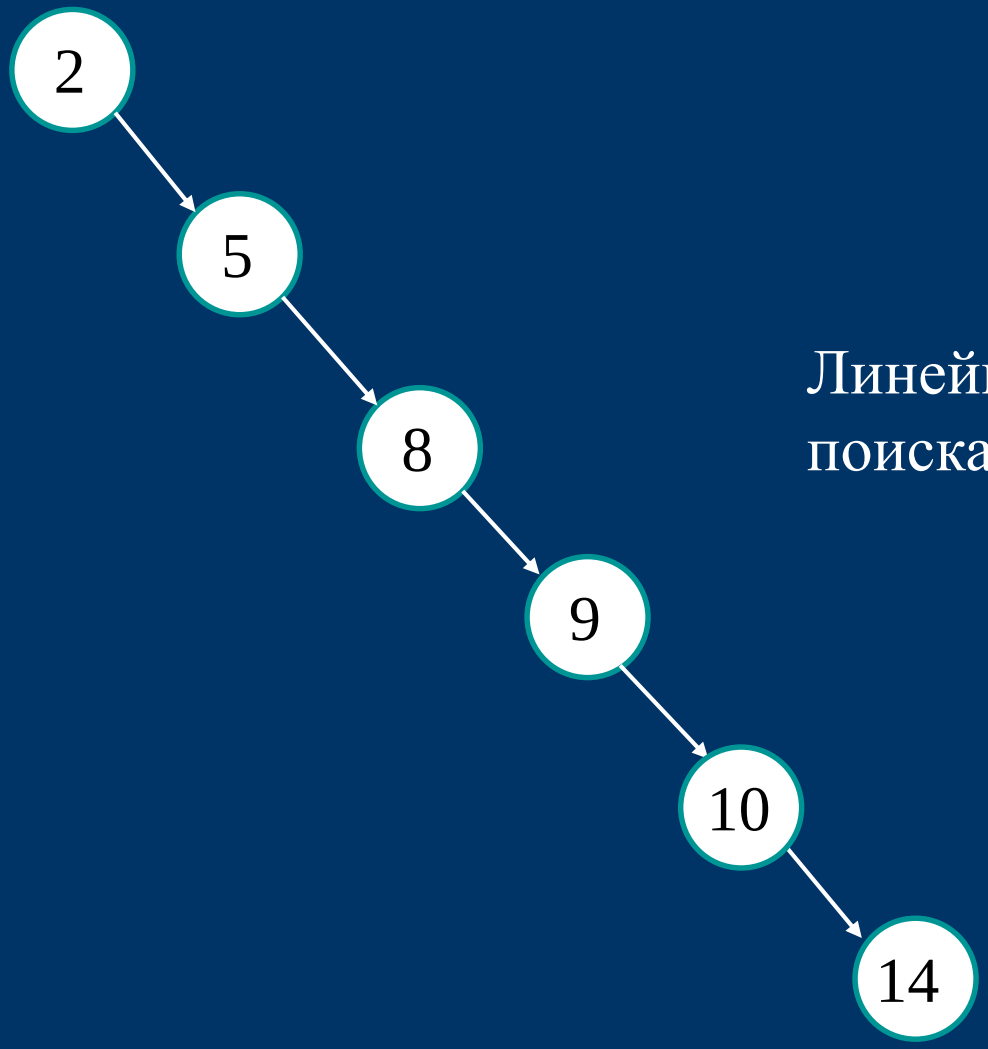


Удаление узла в бинарном дереве



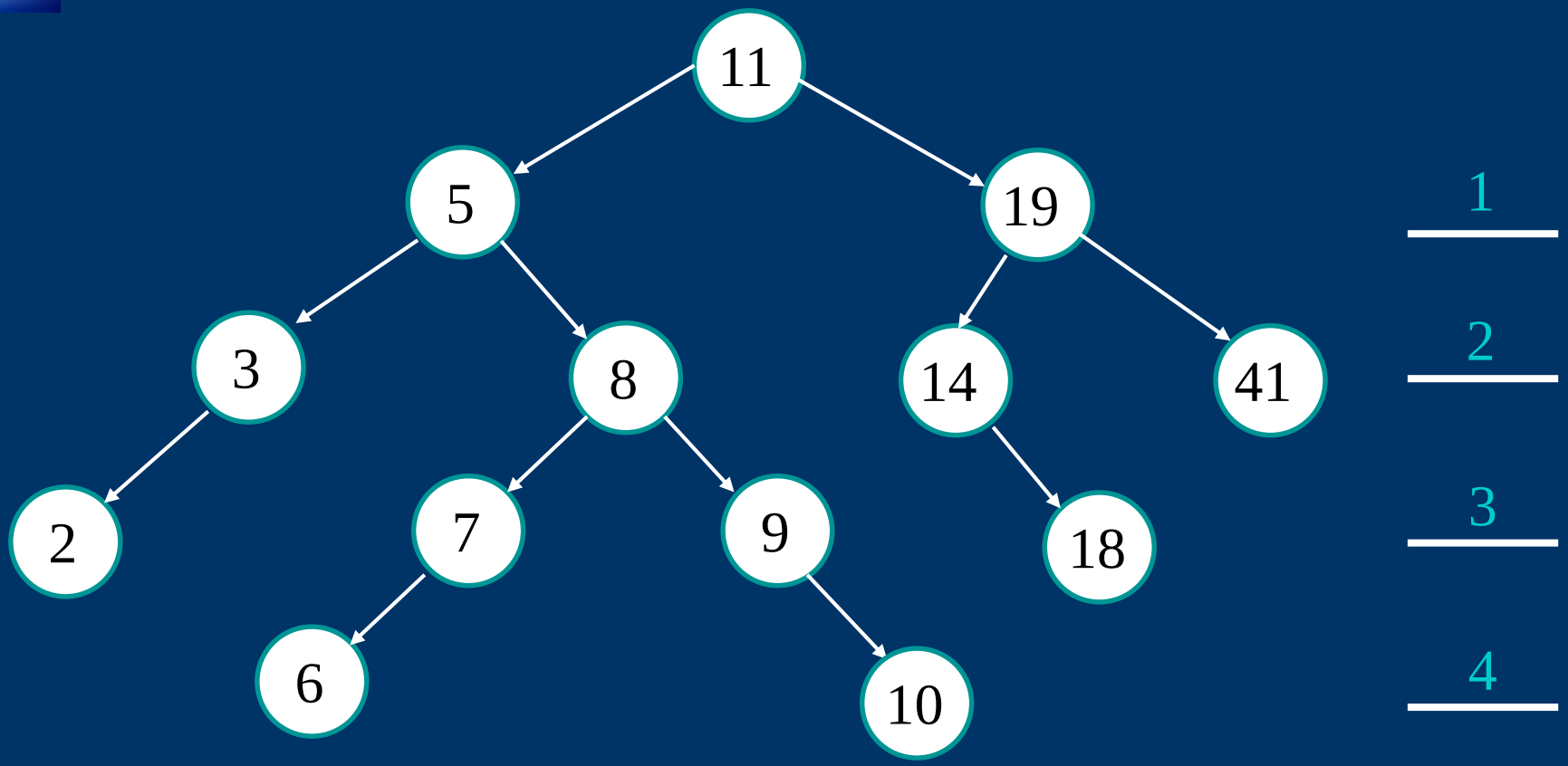
1. Ищем в правом поддереве самый левый лист
2. Меняем местами значения
3. Удаляем лист

Пример неудачного заполнения бинарного дерева



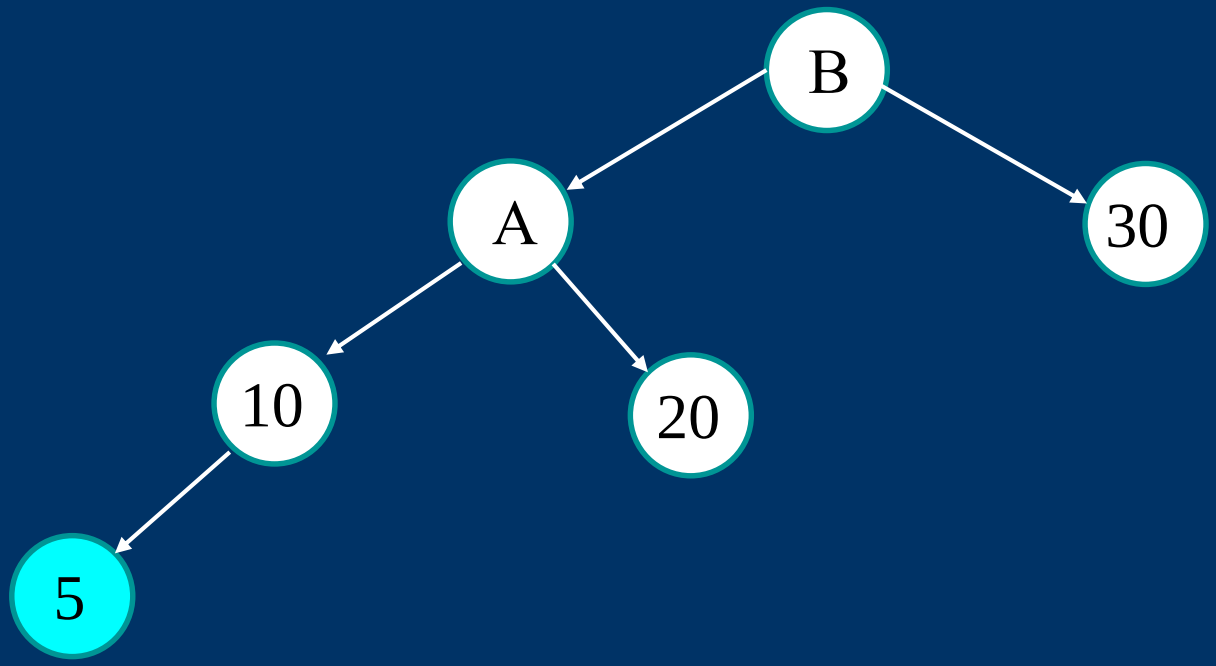
Линейное время
поиска и вставки!

Сбалансированное бинарное дерево



[идеально] Сбалансированное дерево: для любого узла высота левого и правого поддеревьев отличается не больше чем на 1.

Балансировка бинарного дерева



0

1

3

Литература по лекционному материалу

1. Н. Вирт. Алгоритмы и структуры данных. — М.: ДМК Пресс, 2014, 272 с.
2. В.А. Антонюк, А.П. Иванов. Программирование и информатика. - М.: Физический факультет МГУ им. М.В. Ломоносова, 2015.
3. Б. Керниган, Д. Ритчи. Язык программирования С. — М.: Вильямс, 2015, 288 с.