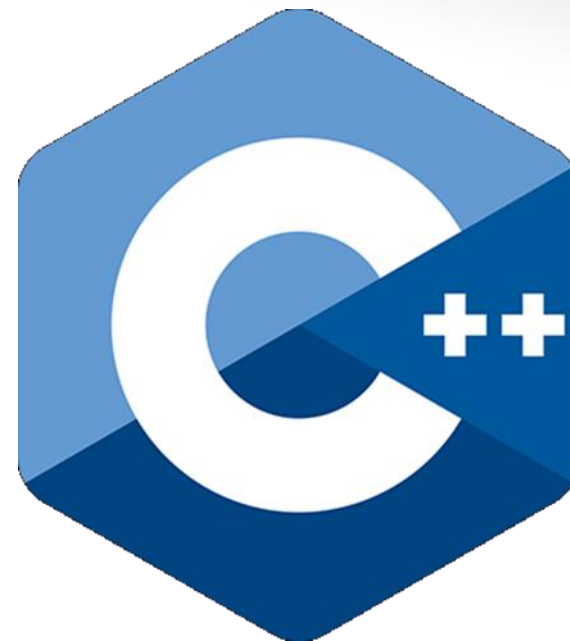


Алгоритмизация и программирование

Программирование на C/C++

(ч.16 – структура данных
«Бинарное дерево»)



Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
<http://www.berkut.mk.ua>

Какие бывают деревья?



pine



birch



willow



maple



oak



ash



elm



linden



spruce

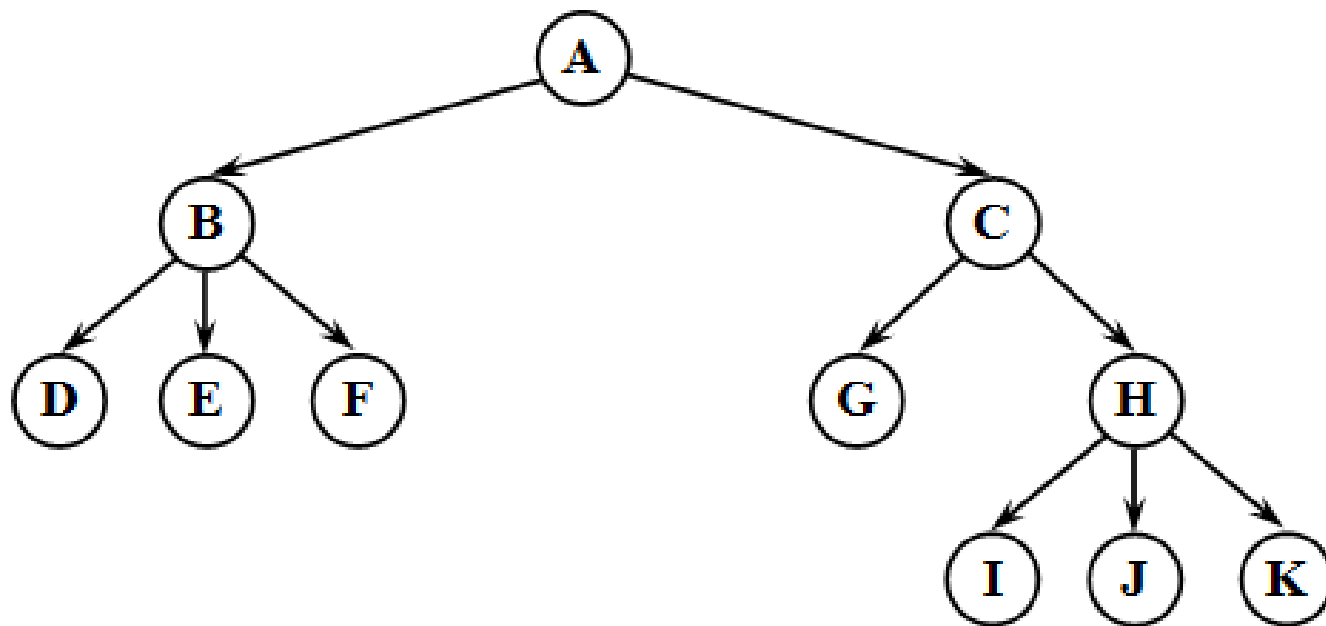
Определения:

- **Дерево** — это динамическая структура данных, состоящая из узлов (вершин), каждый из которых содержит, кроме данных, несколько (две или более) ссылок на различные бинарные деревья.
- На каждый узел имеется ровно одна *ссылка*.
- Начальный узел называется *корнем дерева*.

Строгое определение (по Н.Вирту):

- **Дерево** с базовым типом T – это:
 1. Пустое дерево, либо
 2. Некоторая вершина (узел) типа T , с конечным числом связанных с ней отдельных деревьев с базовым типом T , называемых *поддеревьями*
- Можно сказать, что последовательность (**список**) – это дерево, в котором каждая вершина (узел) имеет не более одного поддерева.
- Поэтому, последовательность (**список**) иногда называют **вырожденным** деревом.

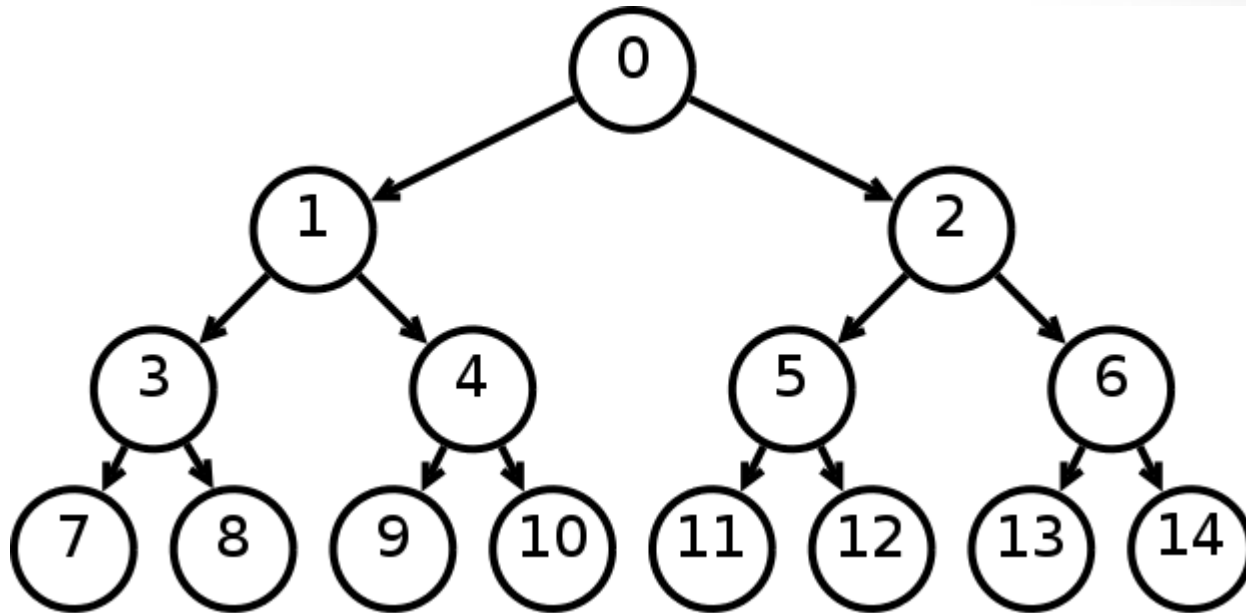
Пример дерева:



Строгое определение (по Н.Вирту):

- **Дерево** с базовым типом T – это:
 1. Пустое дерево, либо
 2. Некоторая вершина (узел) типа T , с конечным числом связанных с ней отдельных деревьев с базовым типом T , называемых *поддеревьями*
- **Бинарное дерево** – дерево, каждая вершина (узел) которого связана не более, чем с двумя поддеревьями.

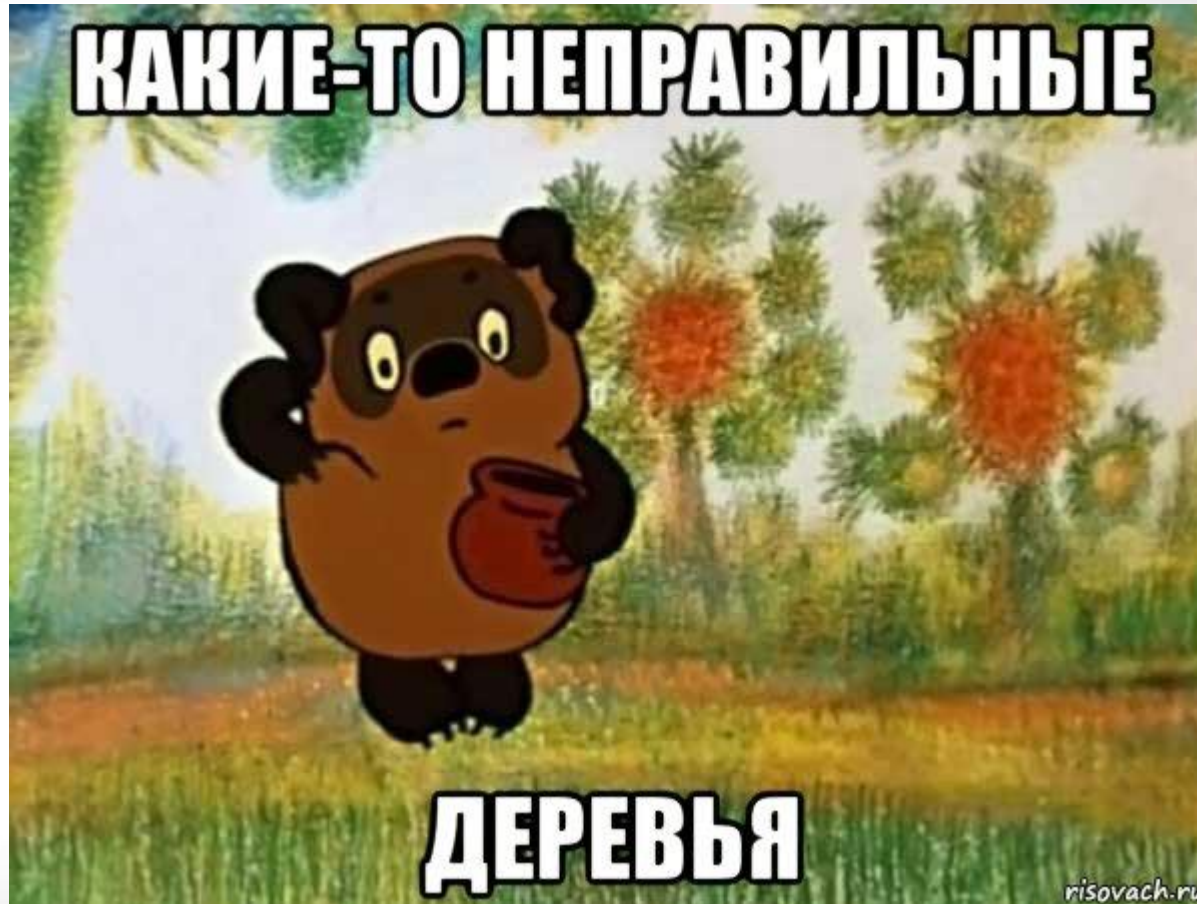
Бинарные деревья



А ещё бывают деревья поиска



Бинарные деревья



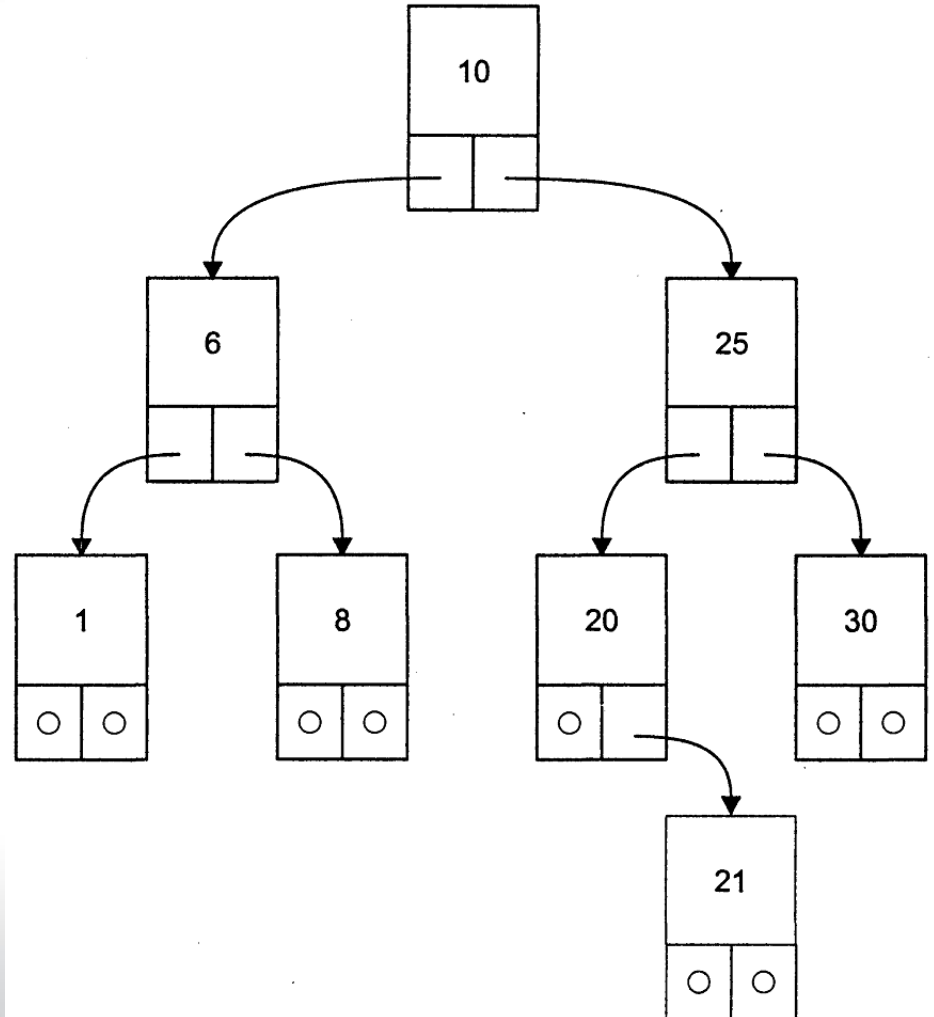
Еще определения

- Узел, не имеющий поддеревьев, называется **листом**.
- Исходящие узлы называются **предками**, входящие — **потомками**.
- Узел, не имеющий предка, называется **корнем**.
- **Высота дерева** определяется количеством уровней, на которых располагаются его узлы.

Бинарные деревья поиска

Если дерево организовано таким образом, что для каждого узла:

- все ключи его левого поддерева меньше ключа этого узла,
- а все ключи его правого поддерева — больше, оно называется **деревом поиска**.
- Одинаковые ключи не допускаются.



Основные операции с бинарным деревом

1. Добавление элемента / создание нового узла
2. Обход дерева
3. Поиск узла по значению ключа в нем
4. Удаление элемента (узла)



1. Добавление элемента / СОЗДАНИЕ НОВОГО УЗЛА

```
bool addElement(int value, node **pNode) {
    if (*pNode == NULL) {
        node *t = new node;
        t->key = value;
        t->left = NULL;
        t->right = NULL;
        *pNode = t;
        return true;
    } else {
        int key = (*pNode)->key;
        if (key == value) return false;
        if (key > value) {
            return addElement(value, &((*pNode)->left));
        } else {
            return addElement(value, &((*pNode)->right));
        }
    }
}
```

РЕКУРСИЯ!

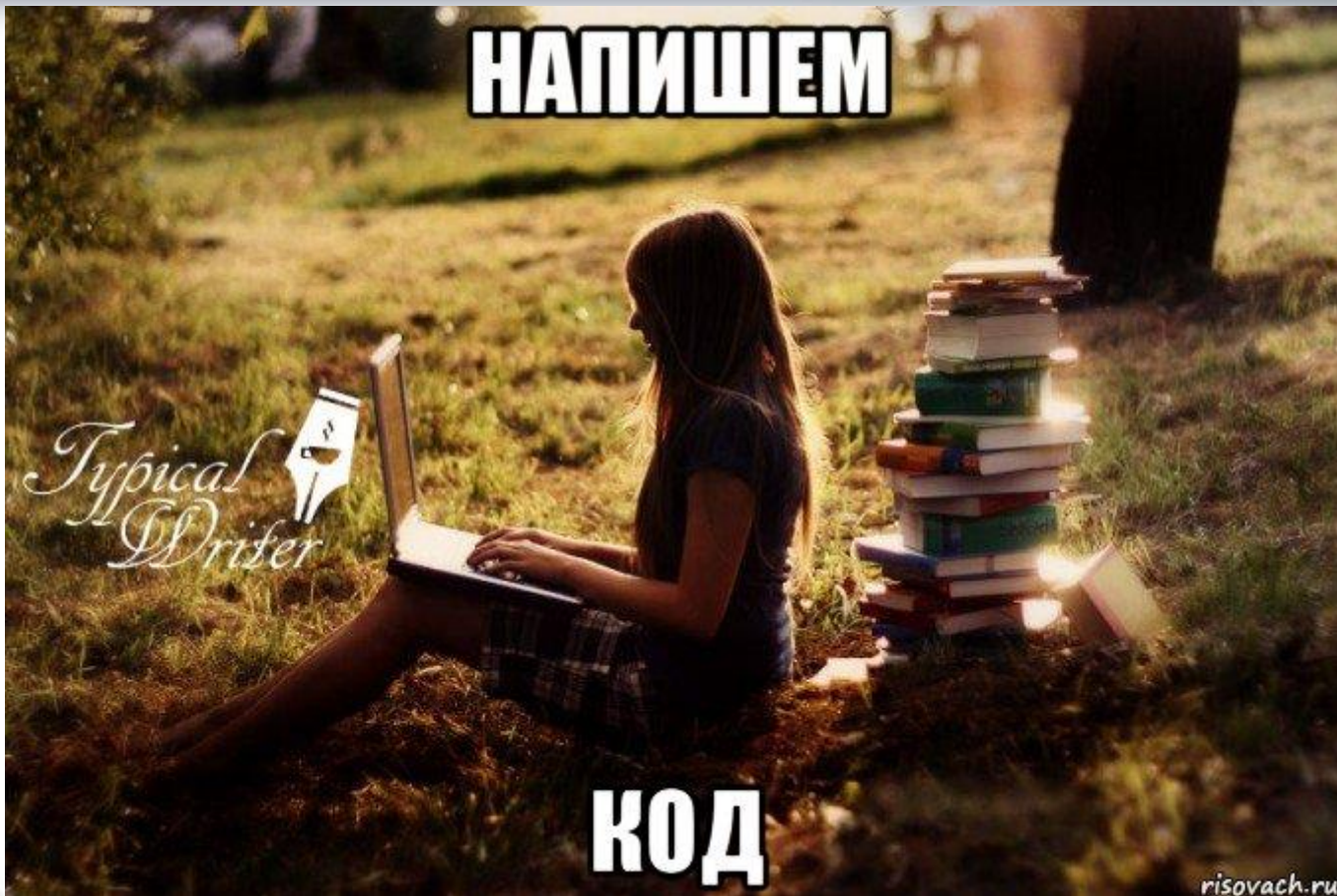
2. Обход дерева

```
void traverseTree (node *pNode) {  
    if (pNode != NULL) {  
        traverseTree (pNode->left);  
        cout << pNode->key << endl;  
        traverseTree (pNode->right);  
    }  
}
```

И снова РЕКУРСИЯ!



Классы и объекты



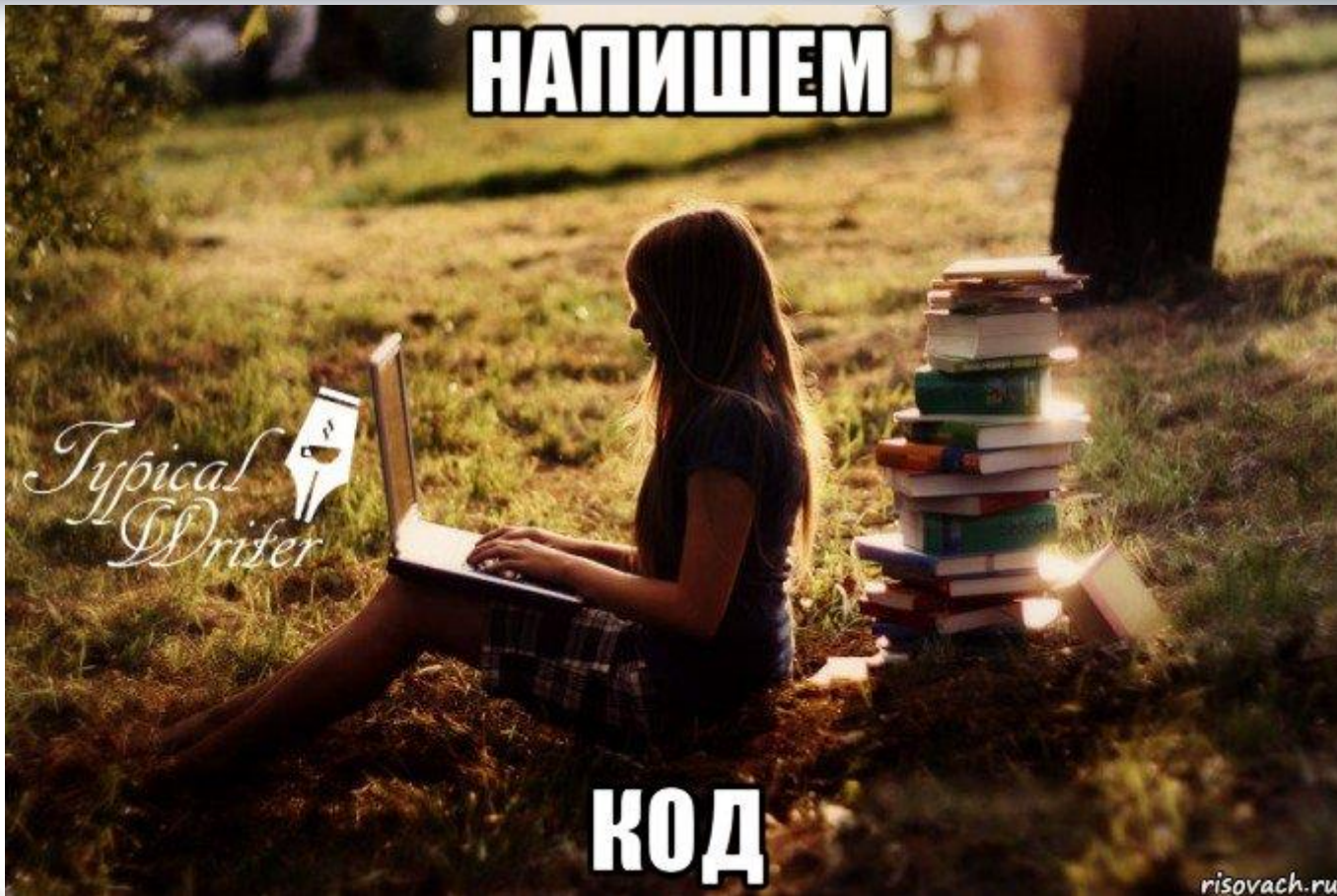
3. Поиск узла по значению ключа

```
node *findElement(int key, node *pNode) {  
    if (pNode == NULL) {  
        return NULL;  
    }  
    if (pNode->key == key) {  
        return pNode;  
    }  
    return findElement(key, (pNode->key > key)  
        ? pNode->left  
        : pNode->right);  
}
```

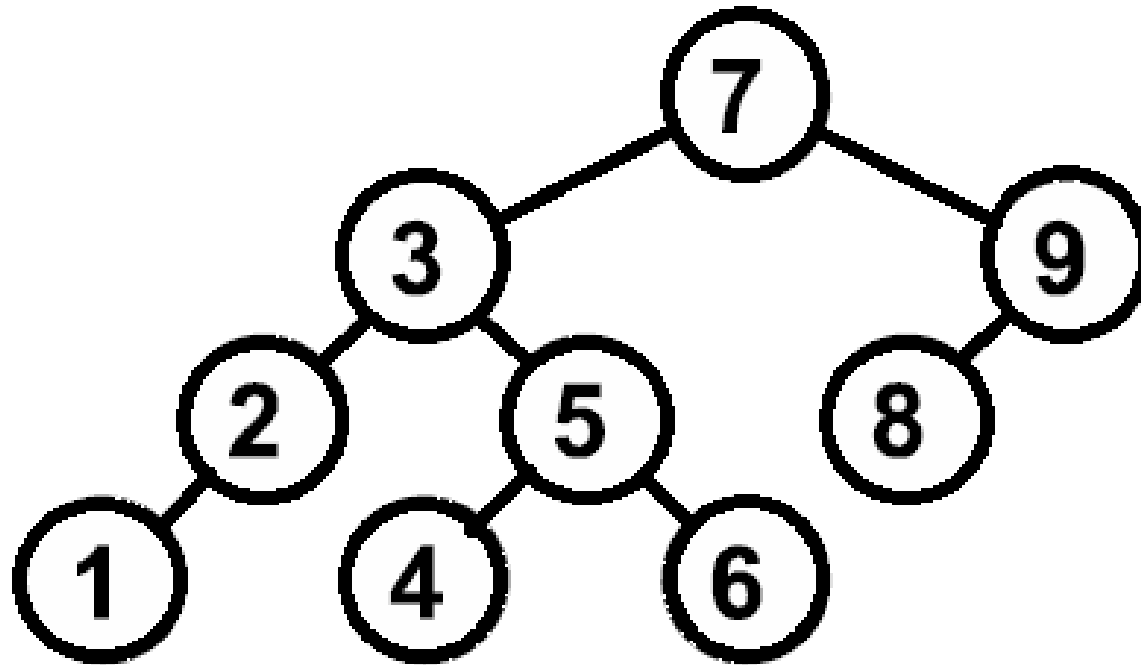
И опять РЕКУРСИЯ!



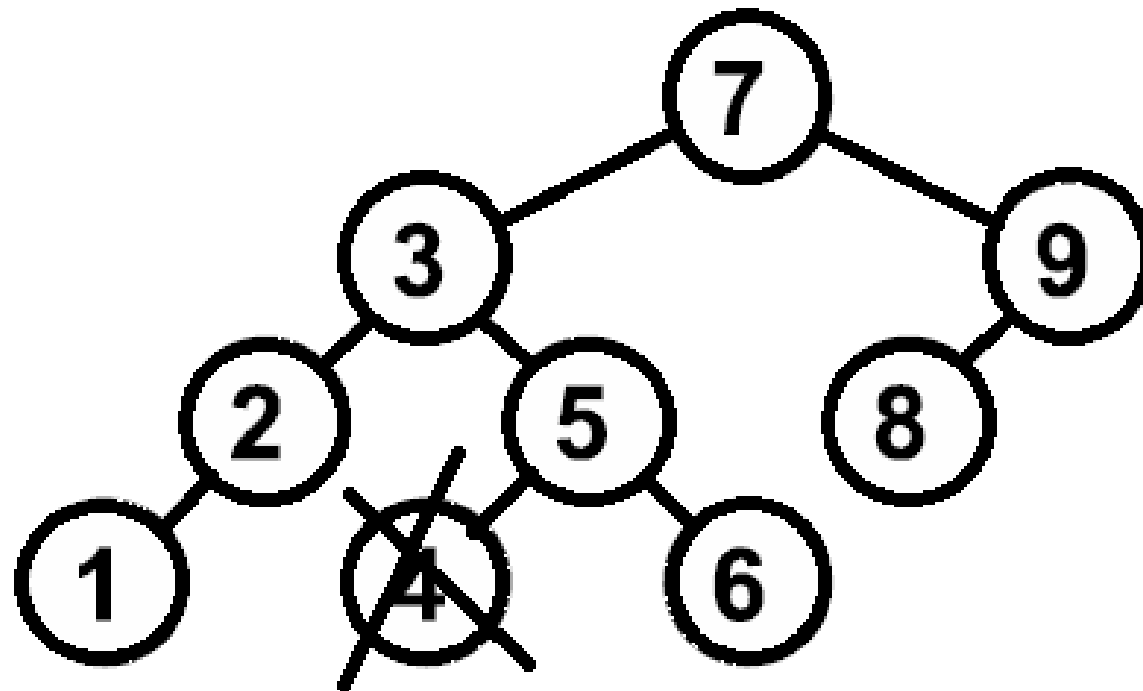
Классы и объекты



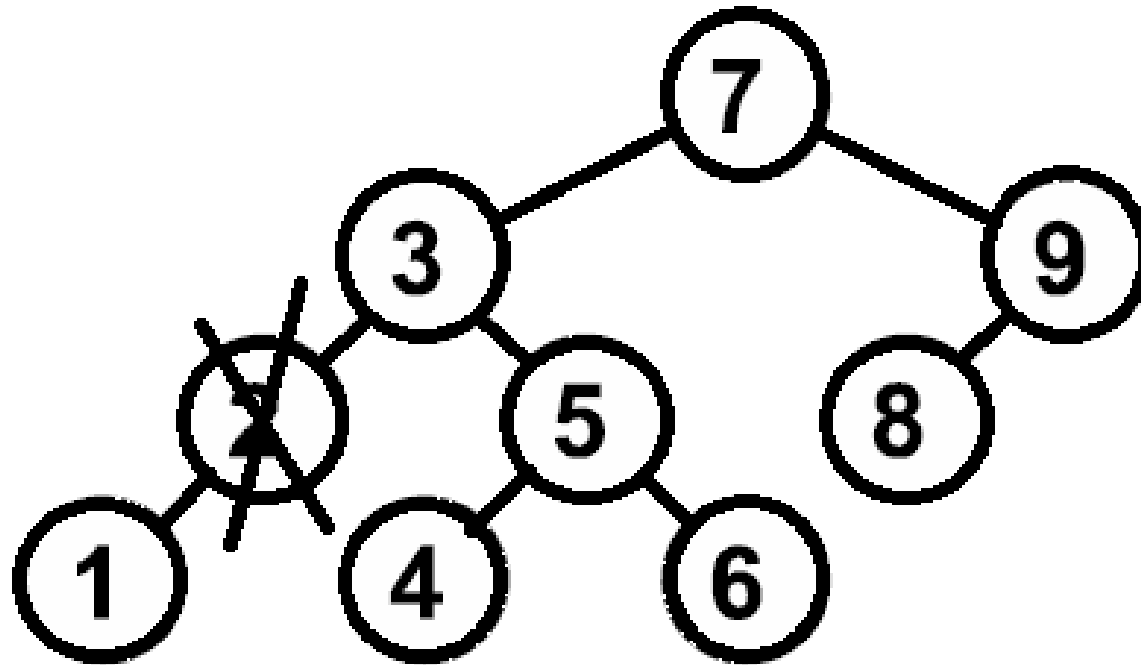
4. Удаление элемента



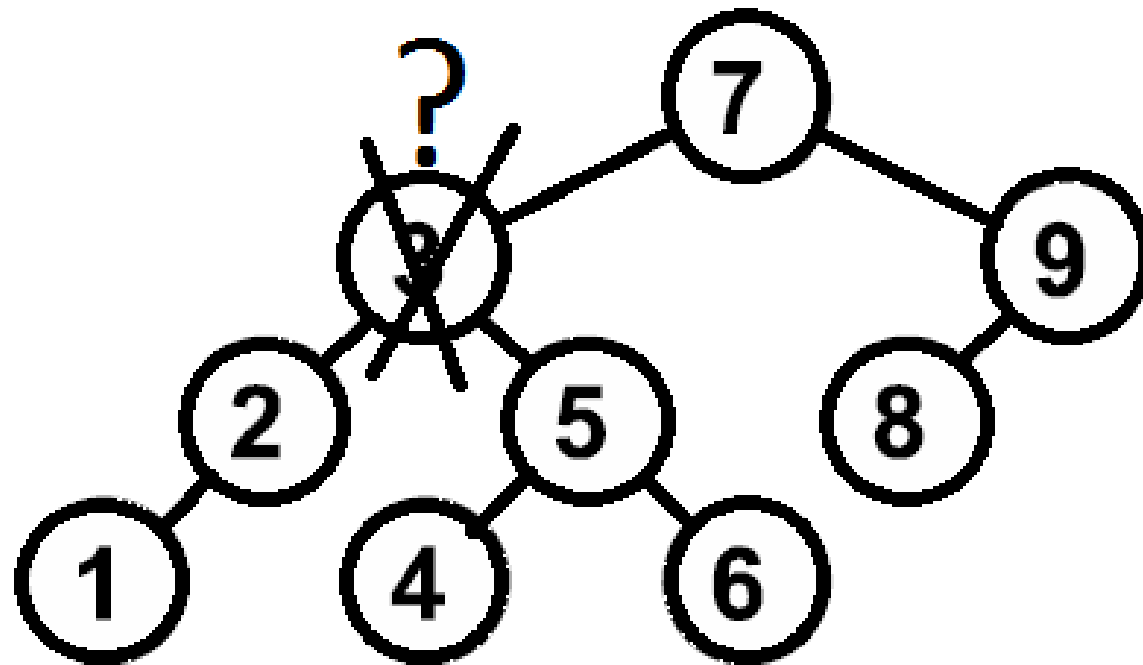
4. Удаление элемента



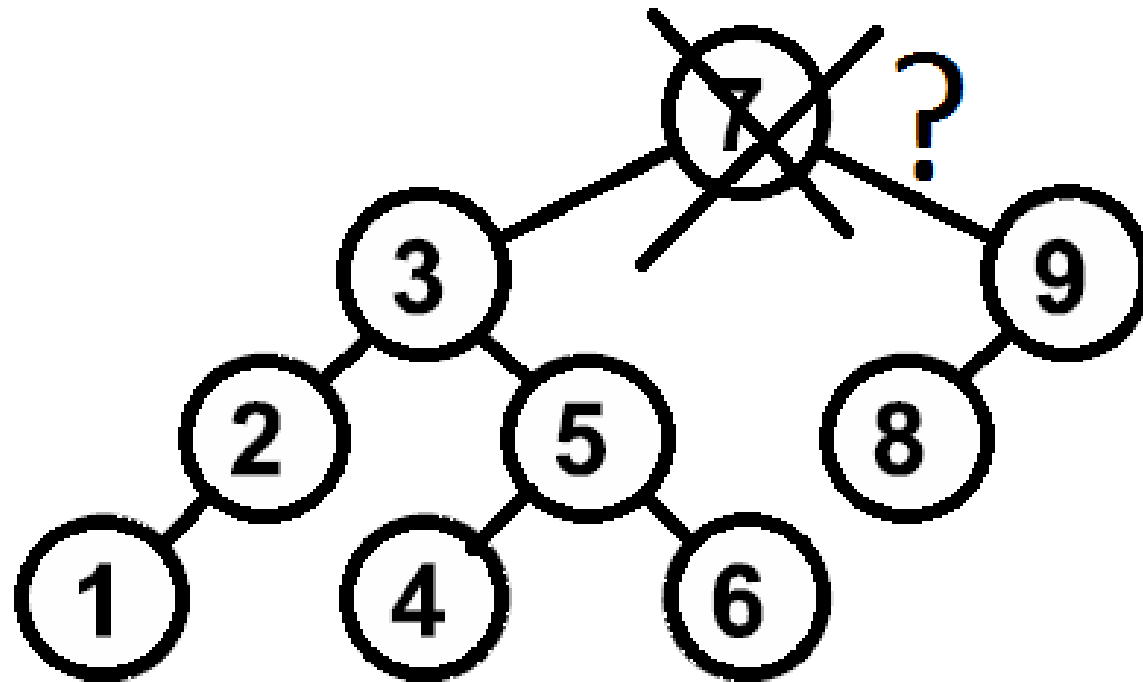
4. Удаление элемента



4. Удаление элемента



4. Удаление элемента



4. Удаление элемента

Удаление элемента – совсем непросто!

Если исключаемый элемент – лист или узел с одним потомком, всё просто.

Если нужно удалить элемент с двумя потомками, получаем проблему!

Удаляемый элемент нужно заменить:

- Либо на самый правый элемент его левого поддеревя,
- Либо на самый левый элемент его правого поддеревя

Причём, он должен иметь не более одного потомка!



4. Удаление элемента

```
void removeElement(int value, node **pNode) {
    if (*pNode == NULL); else {
        if (value < (*pNode)->key)
            removeElement(value, &((*pNode)->left));
        else {
            if (value > (*pNode)->key)
                removeElement(value, &((*pNode)->right));
            else {
                node *q = *pNode;
                if (q->right == NULL) *pNode = q->left;
                else if (q->left == NULL) *pNode = q->right; else {
                    removeEl (&(q->left), &q);
                }
                delete q;
            }
        }
    }
}
```

И опять РЕКУРСИЯ!

Но это еще не всё!

4. Удаление элемента

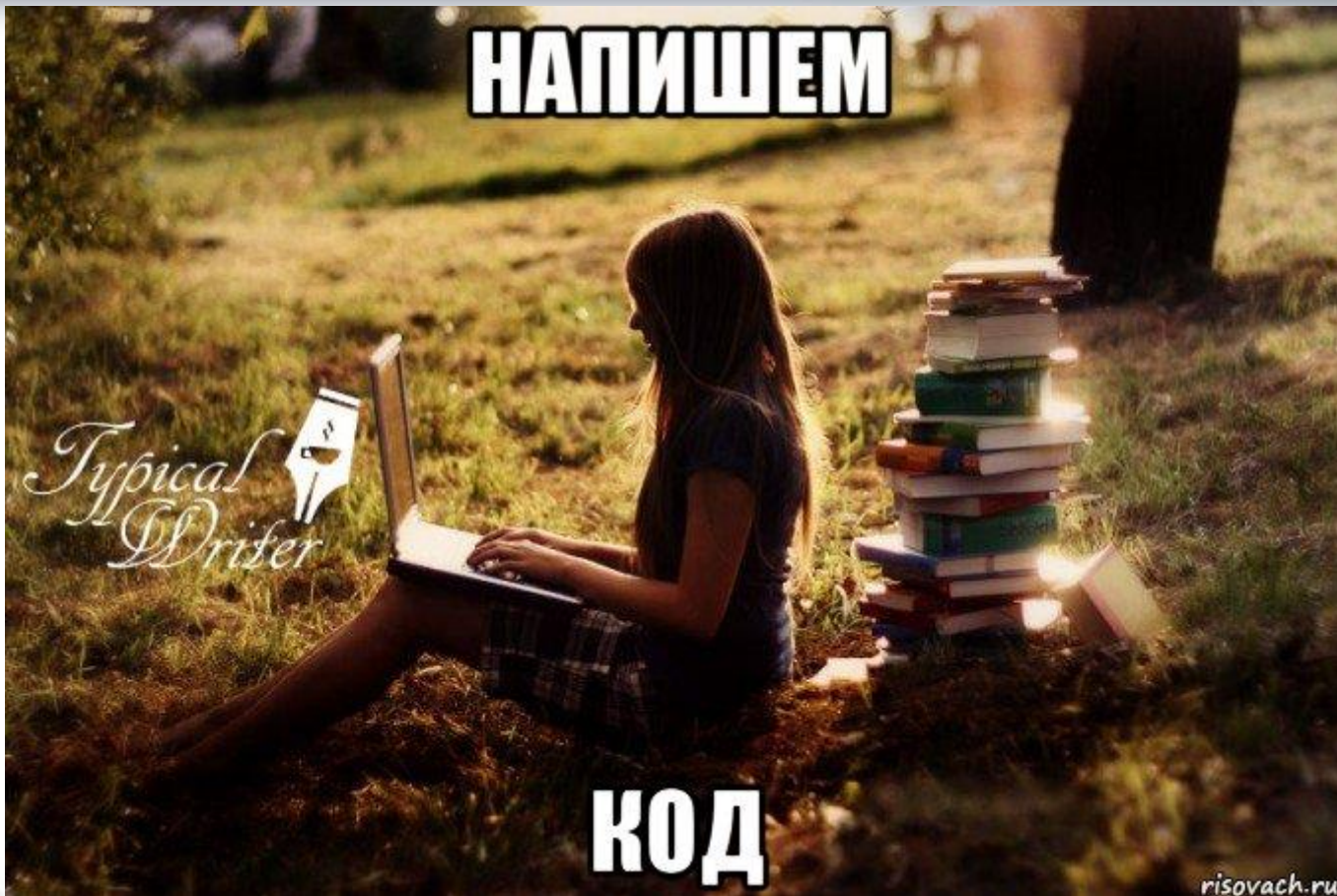
```
void removeEl (node **pNode, node **q) {  
    if ((*pNode)->right != NULL) removeEl (&(*pNode)->right, q);  
    else {  
        (*q)->key = (*pNode)->key;  
        *q = *pNode;  
        *pNode = (*pNode)->left;  
    }  
}
```


Бинарные деревья





Классы и объекты





НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ
КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА



Алгоритмизация и программирование

Программирование на C/C++

(ч.16 – структура данных
«Бинарное дерево»)



Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
<http://www.berkut.mk.ua>