НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Алгоритмизация и программирование

Программирование на C/C++/Kotlin

(ч.9 – структура данных «Бинарное дерево»)

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

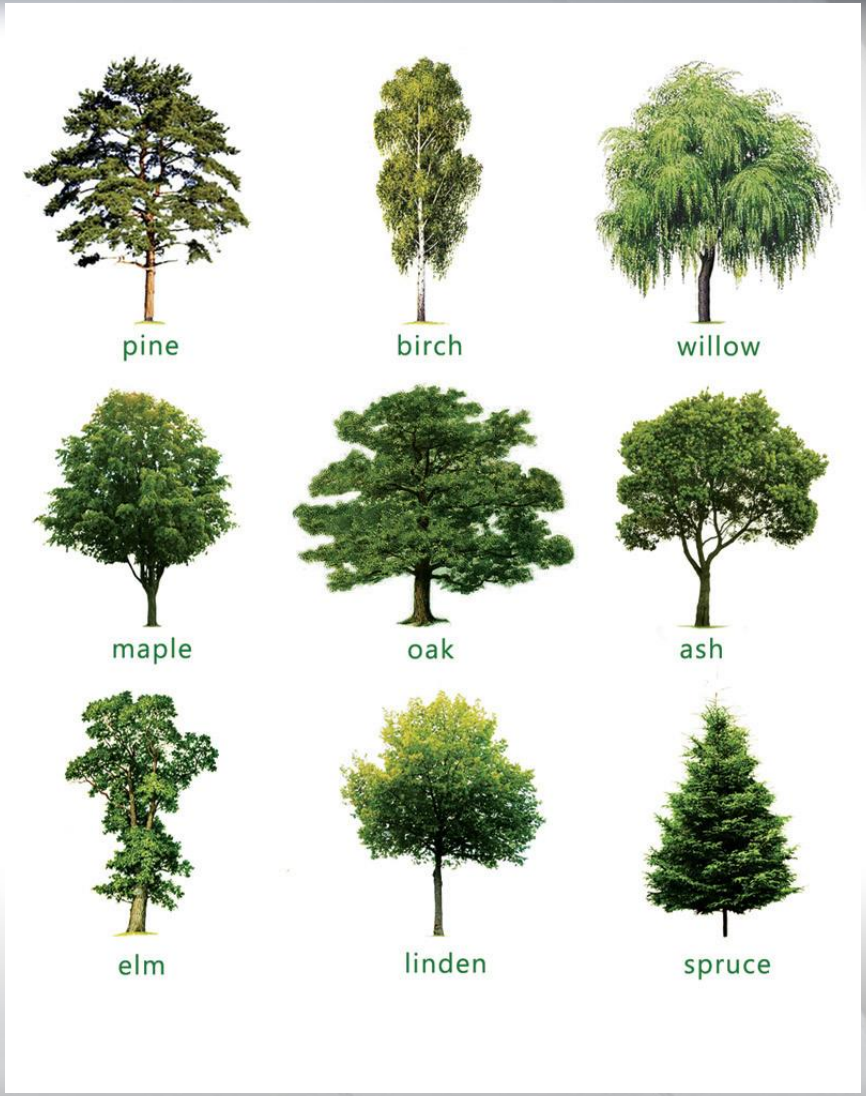
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Деревья

Какие бывают деревья?

Деревья

Какие бывают деревья?



Деревья

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*.

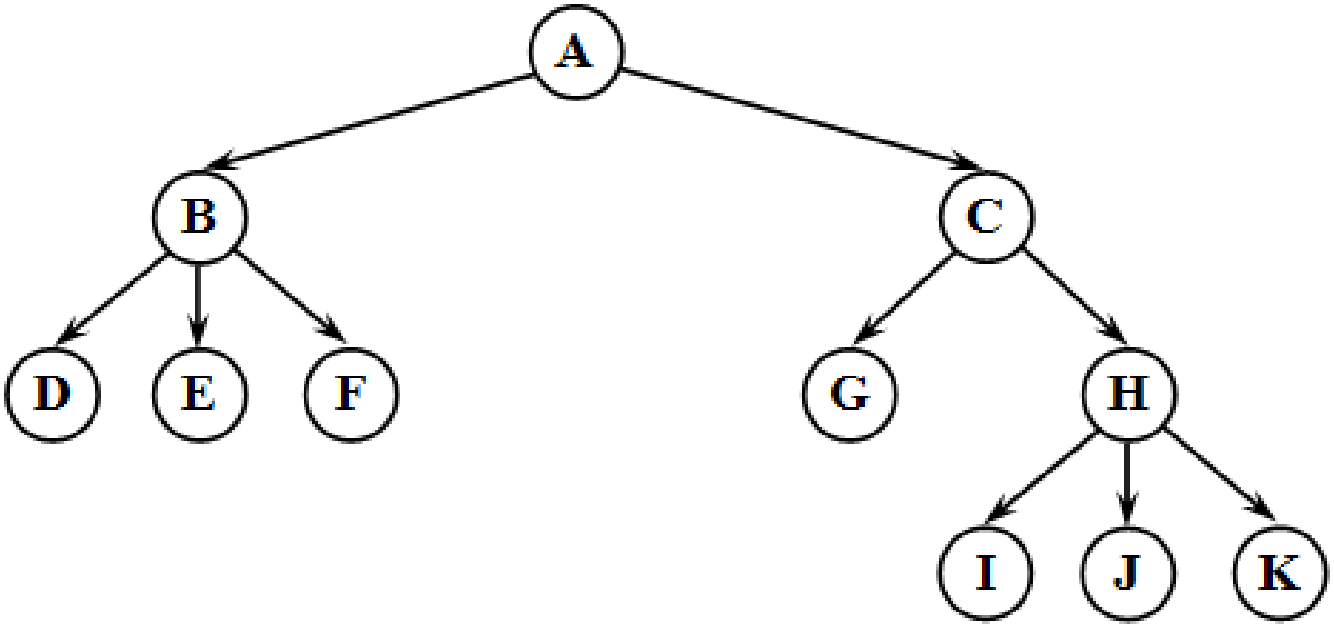
Деревья

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.

Деревья

Пример дерева:

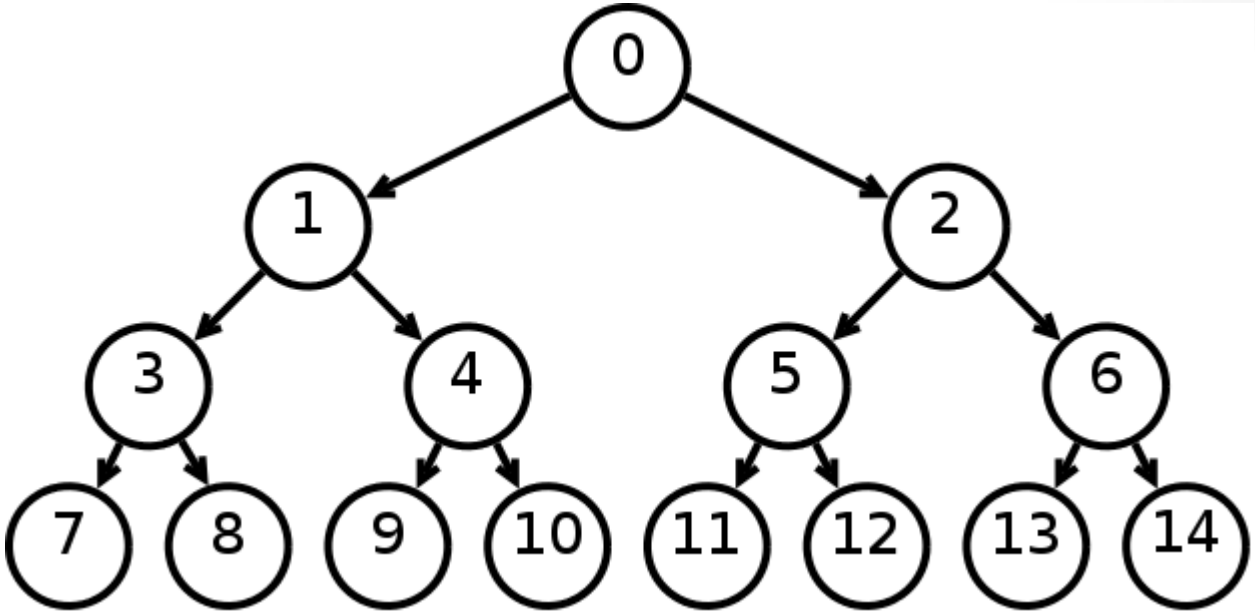


Деревья

Строгое определение (по Н.Вирту):

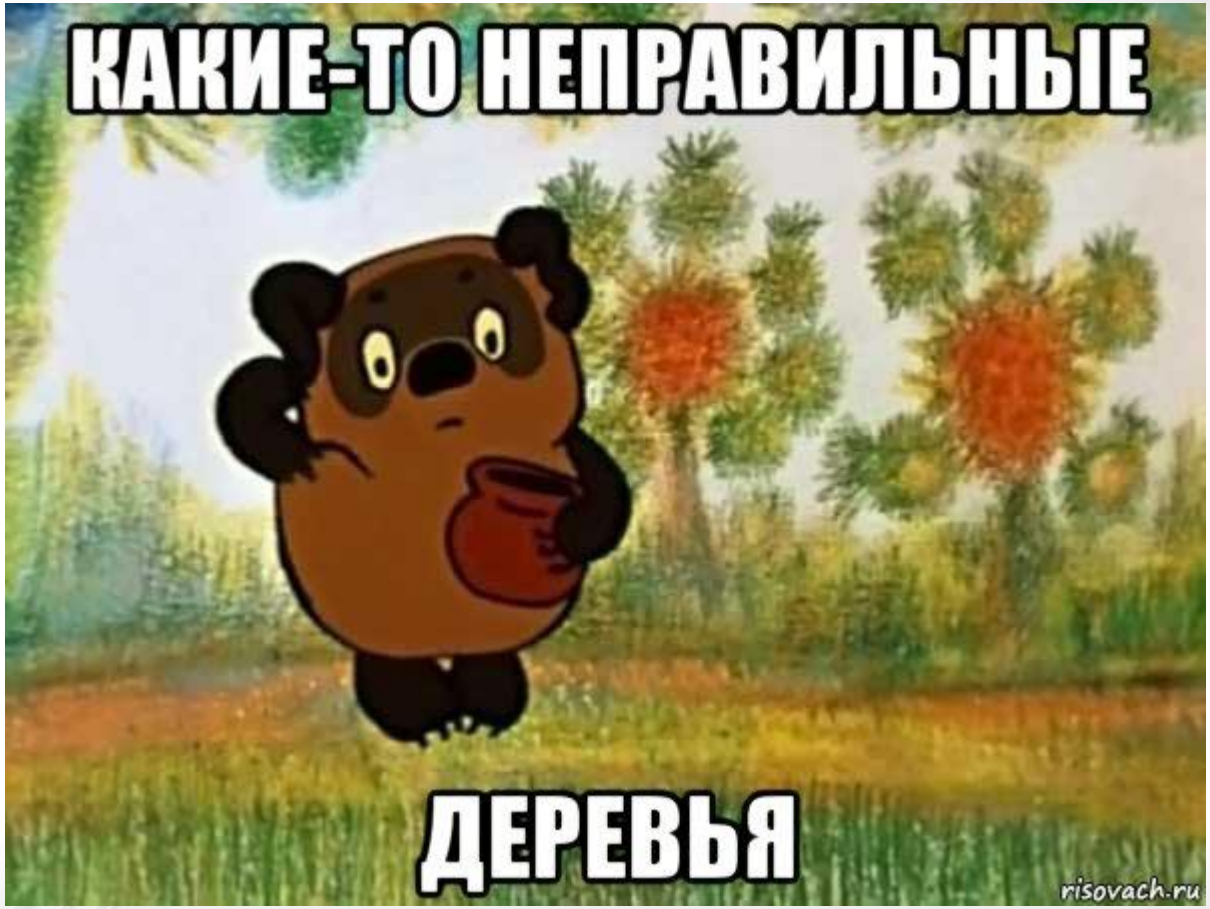
- ***Дерево*** с базовым типом T – это:
  1. Пустое дерево, либо
  2. Некоторая вершина (узел) типа T, с конечным числом связанных с ней отдельных деревьев с базовым типом T, называемых *поддеревьями*
- ***Бинарное дерево*** – дерево, каждая вершина (узел) которого связана не более, чем с двумя поддеревьями.

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.

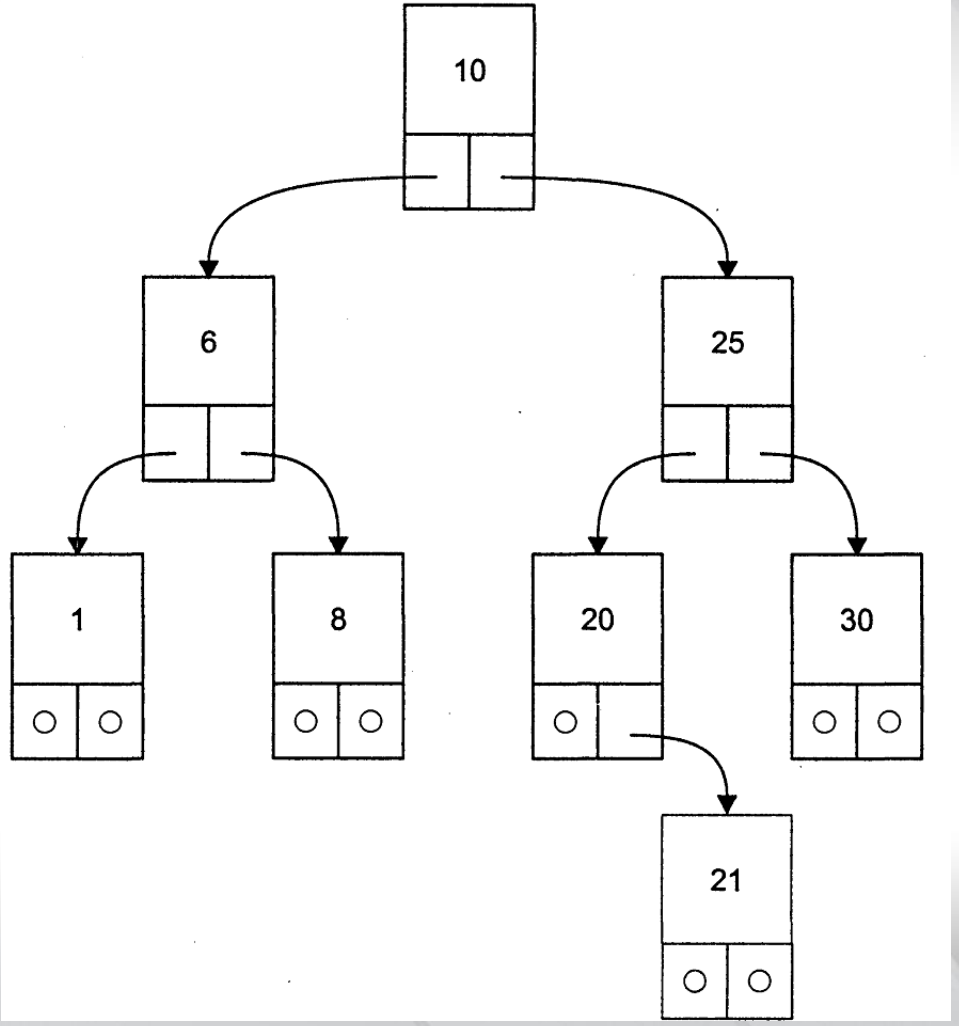
Бинарные деревья поиска

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.



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

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)

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

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));
        }
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

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);
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

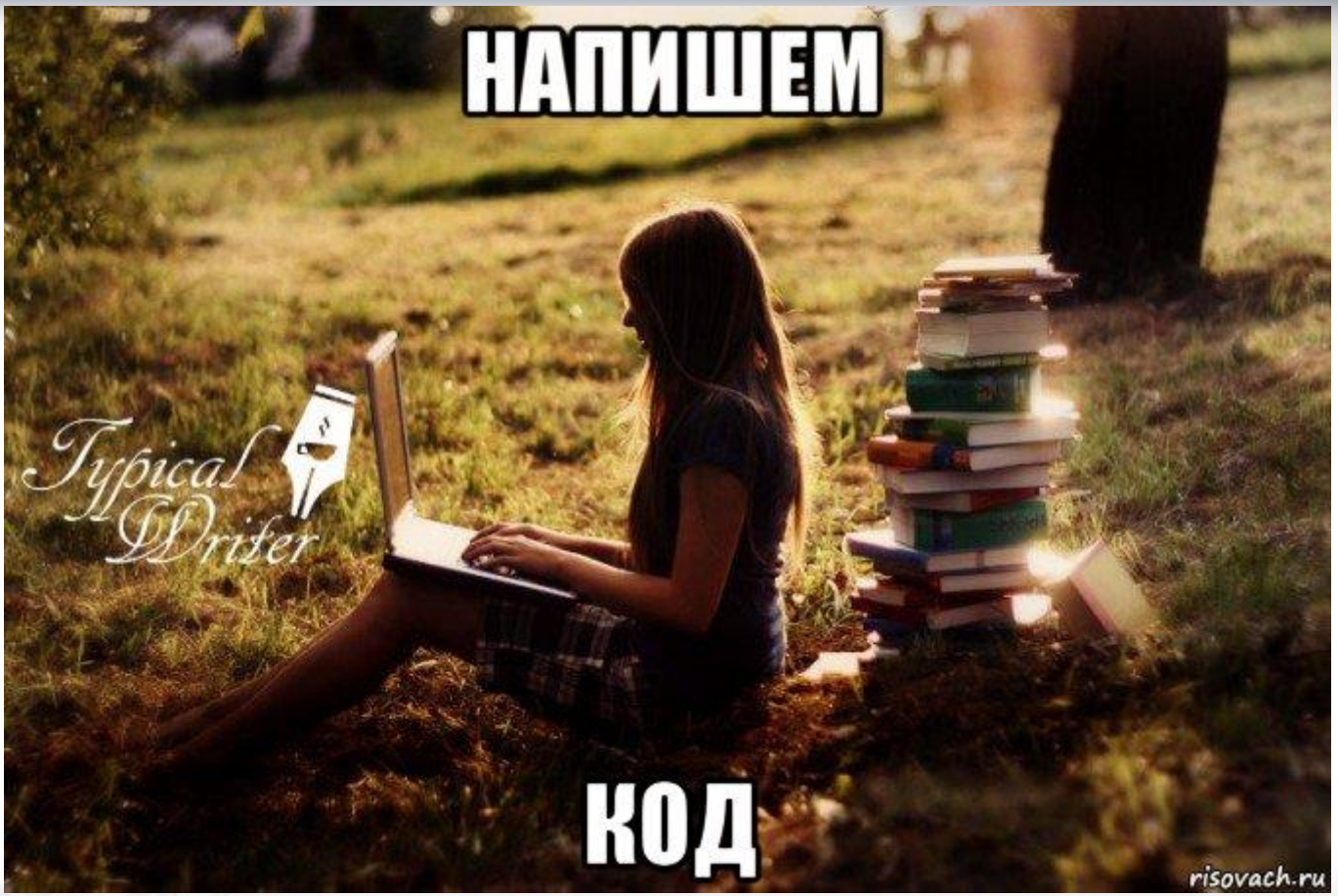
2. Обход дерева

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

И снова РЕКУРСИЯ!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

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);
}
```

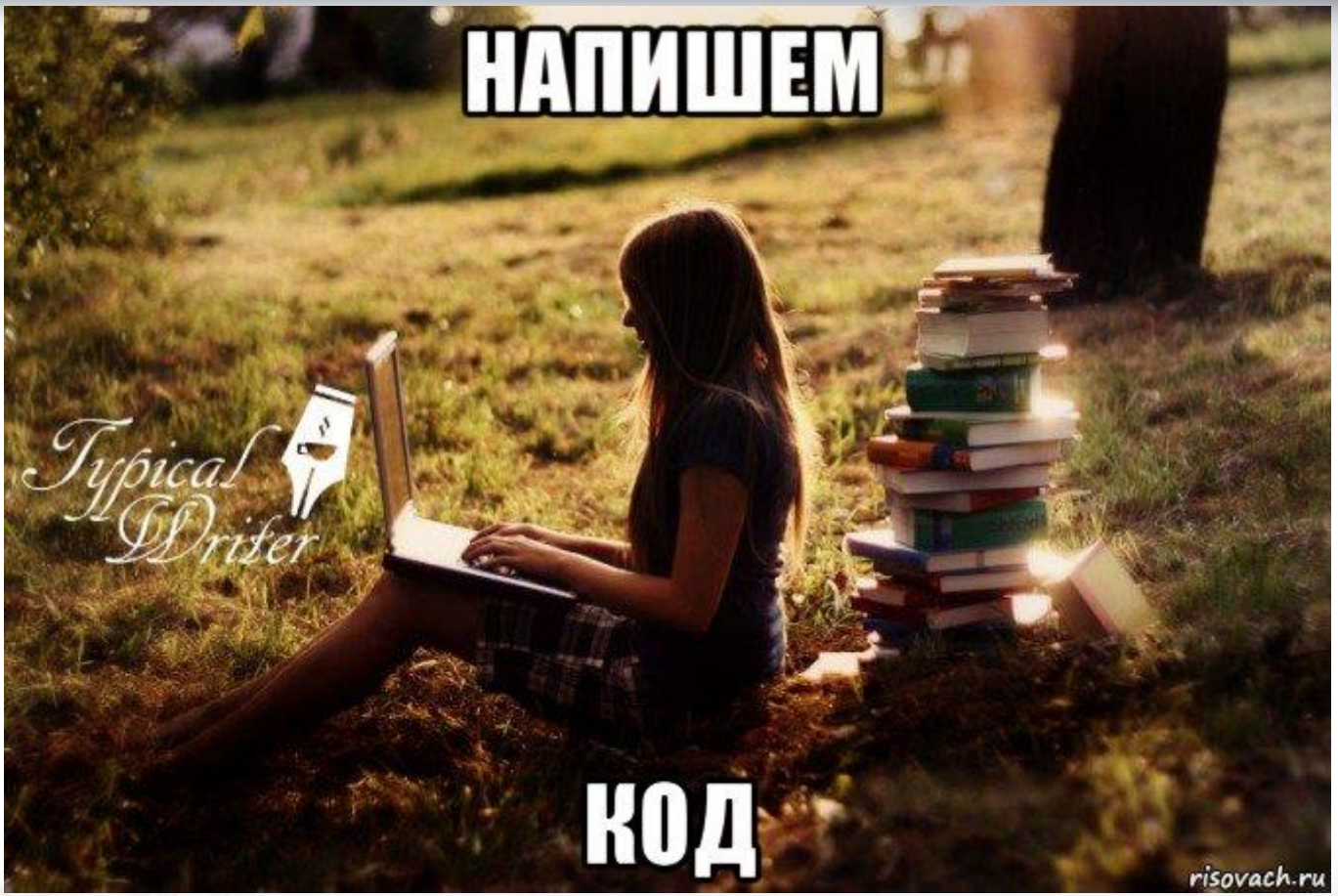
3. Поиск узла по значению ключа

```
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);
}
```

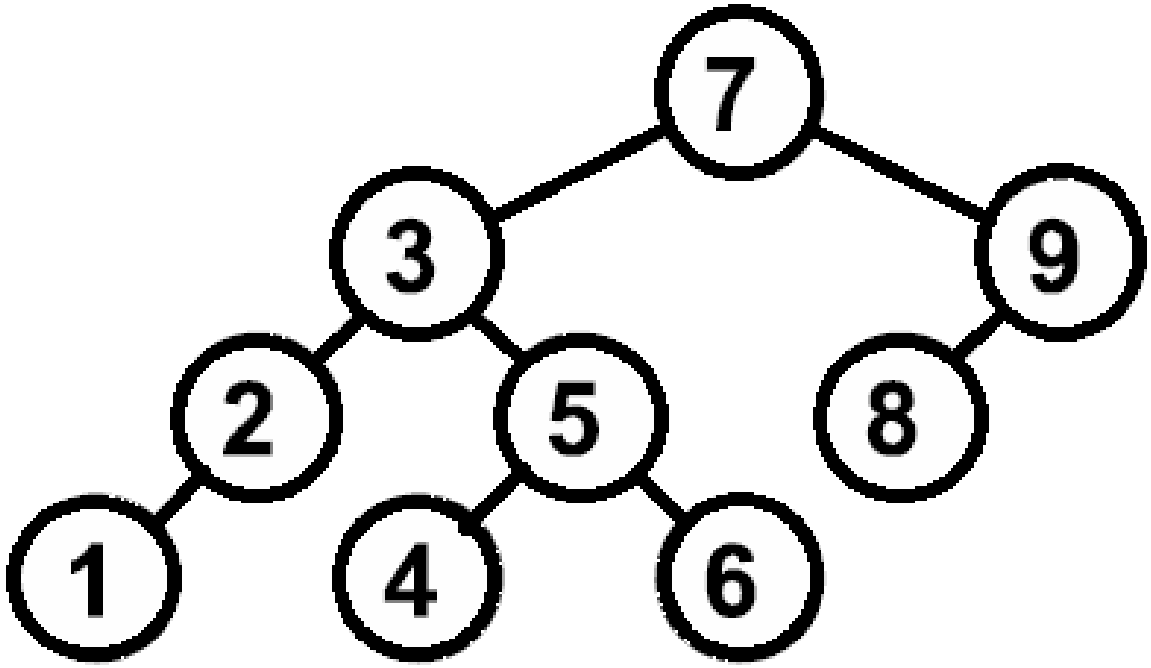
И опять РЕКУРСИЯ!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

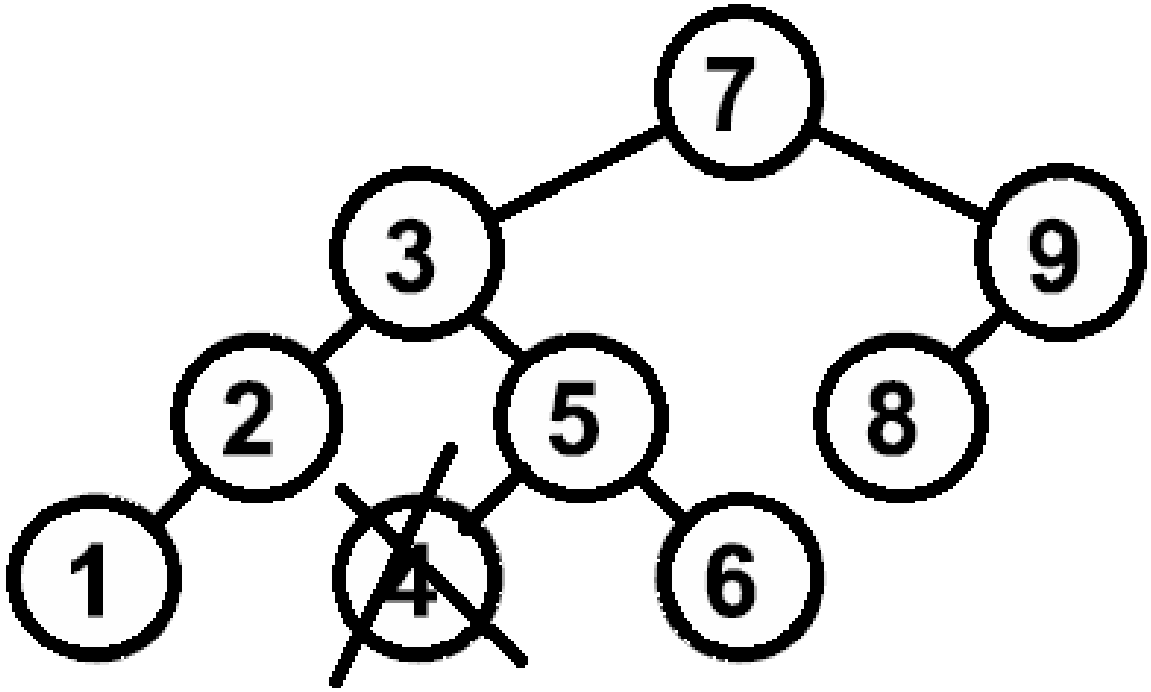
Классы и объекты



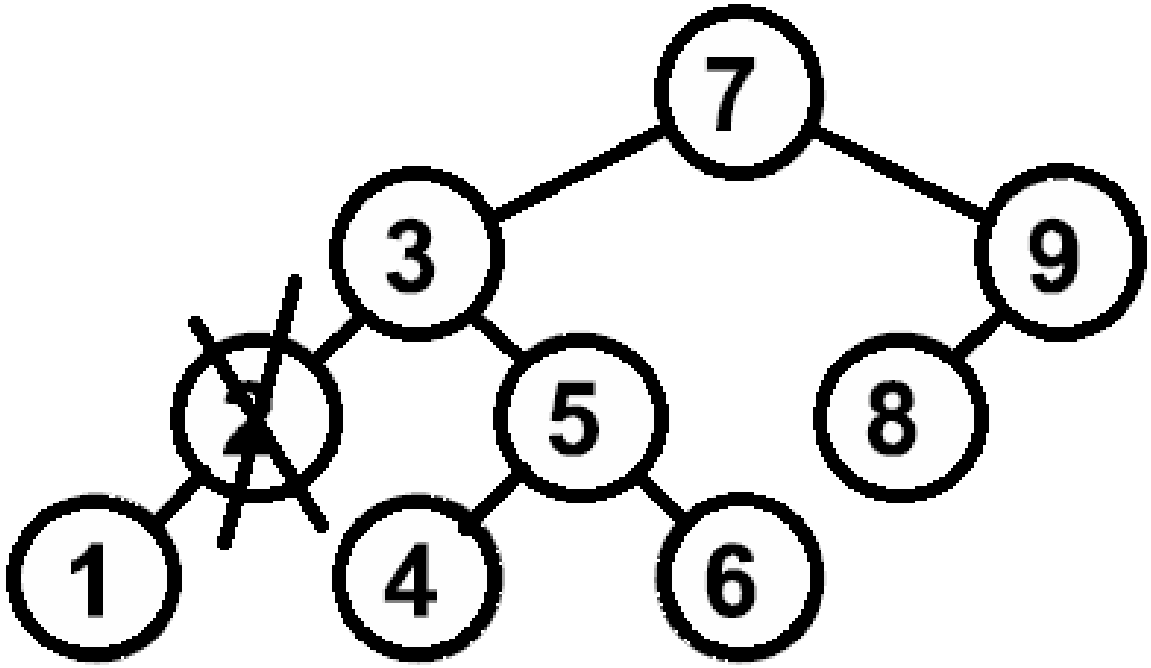
4. Удаление элемента



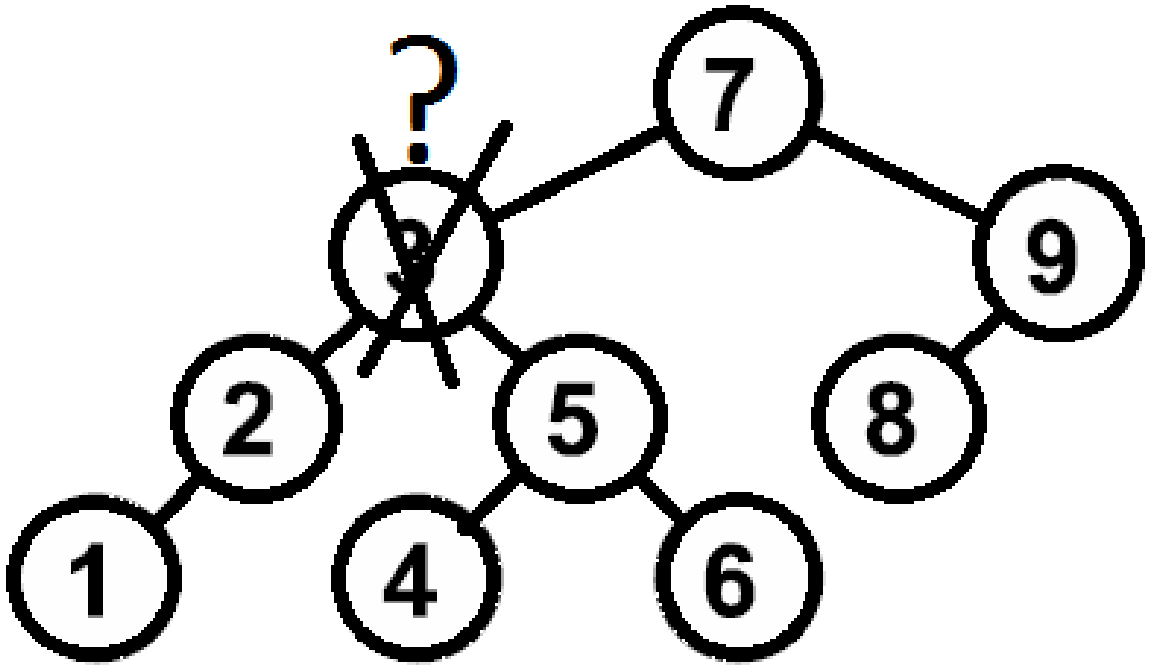
4. Удаление элемента



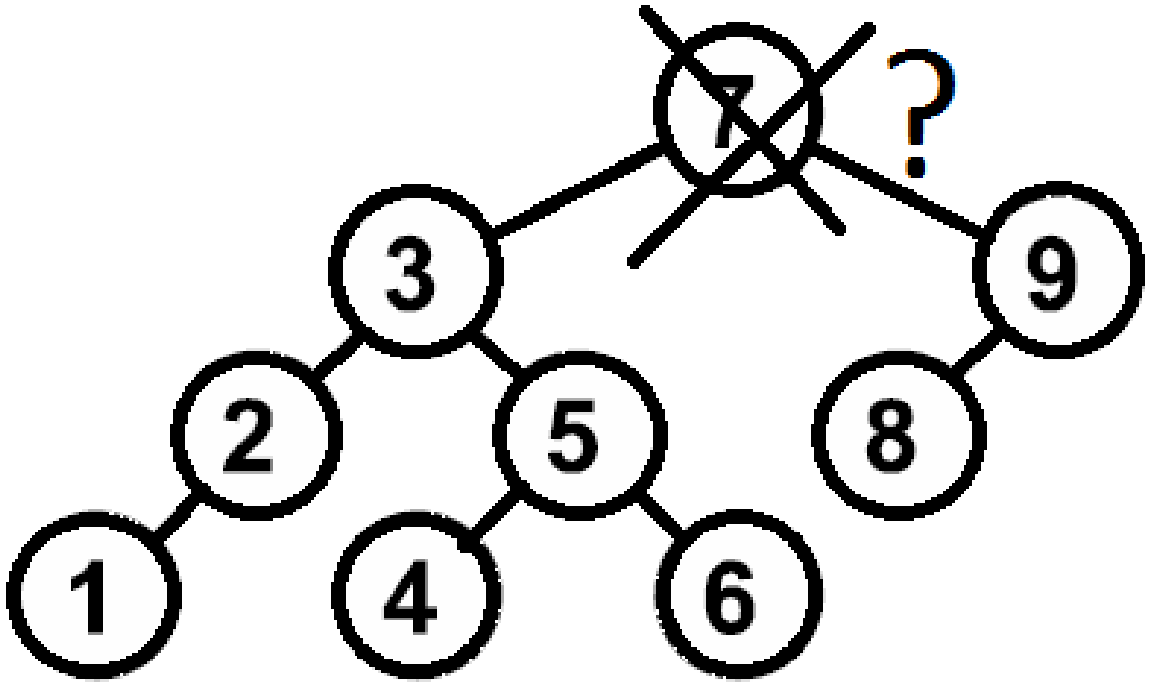
4. Удаление элемента



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!

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 поддерева,
- Либо на самый левый элемент его правого поддерева

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;
            }
        }
    }
}
```

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!

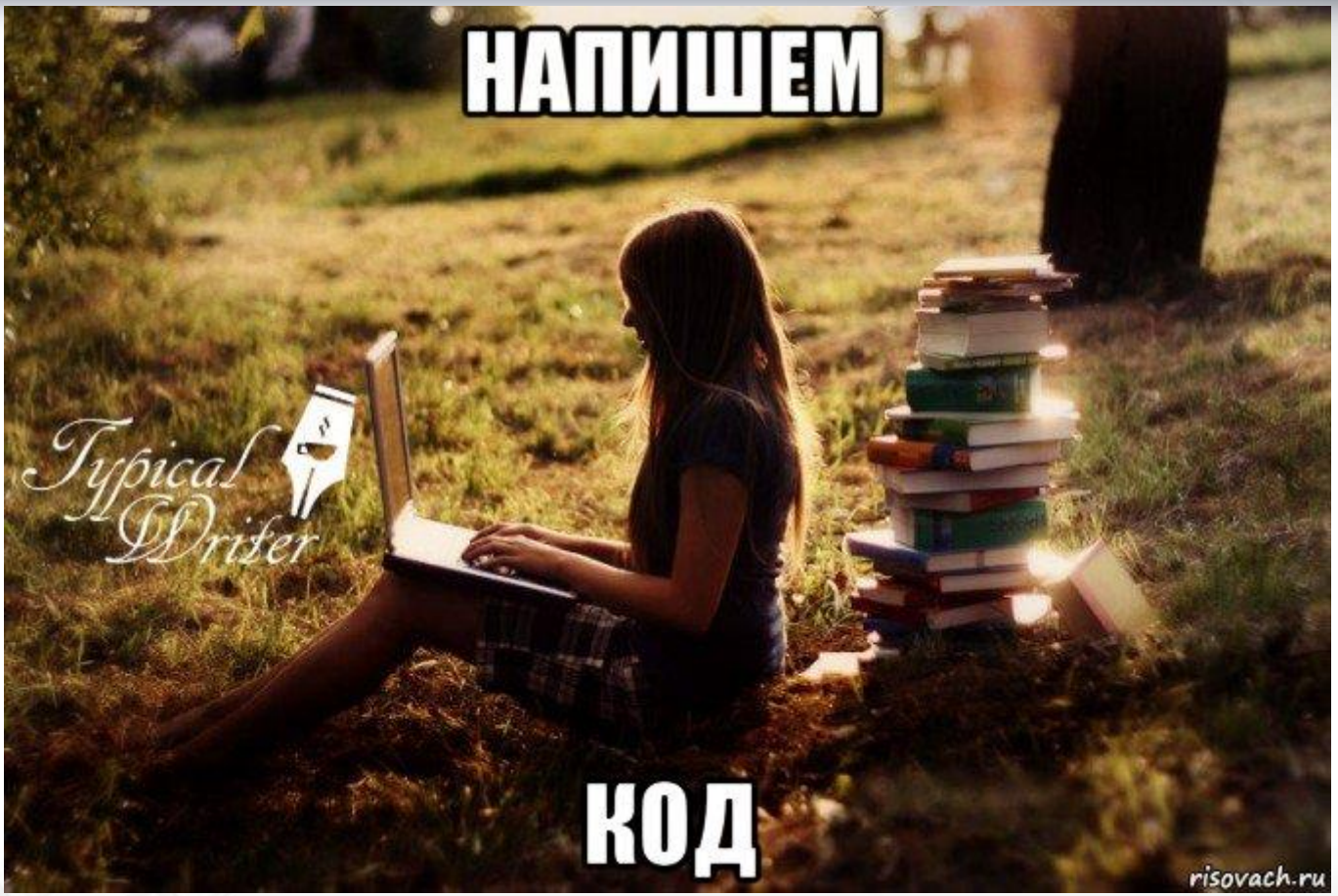
4. Удаление элемента

```
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

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Классы и объекты



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Алгоритмизация и программирование

Программирование на C/C++/Kotlin

(ч.9 – структура данных «Бинарное дерево»)

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