Bài 9
Stack – Queue – Tree

Module: **ADVANCED PROGRAMMING WITH PHP 2.0**

Mục tiêu

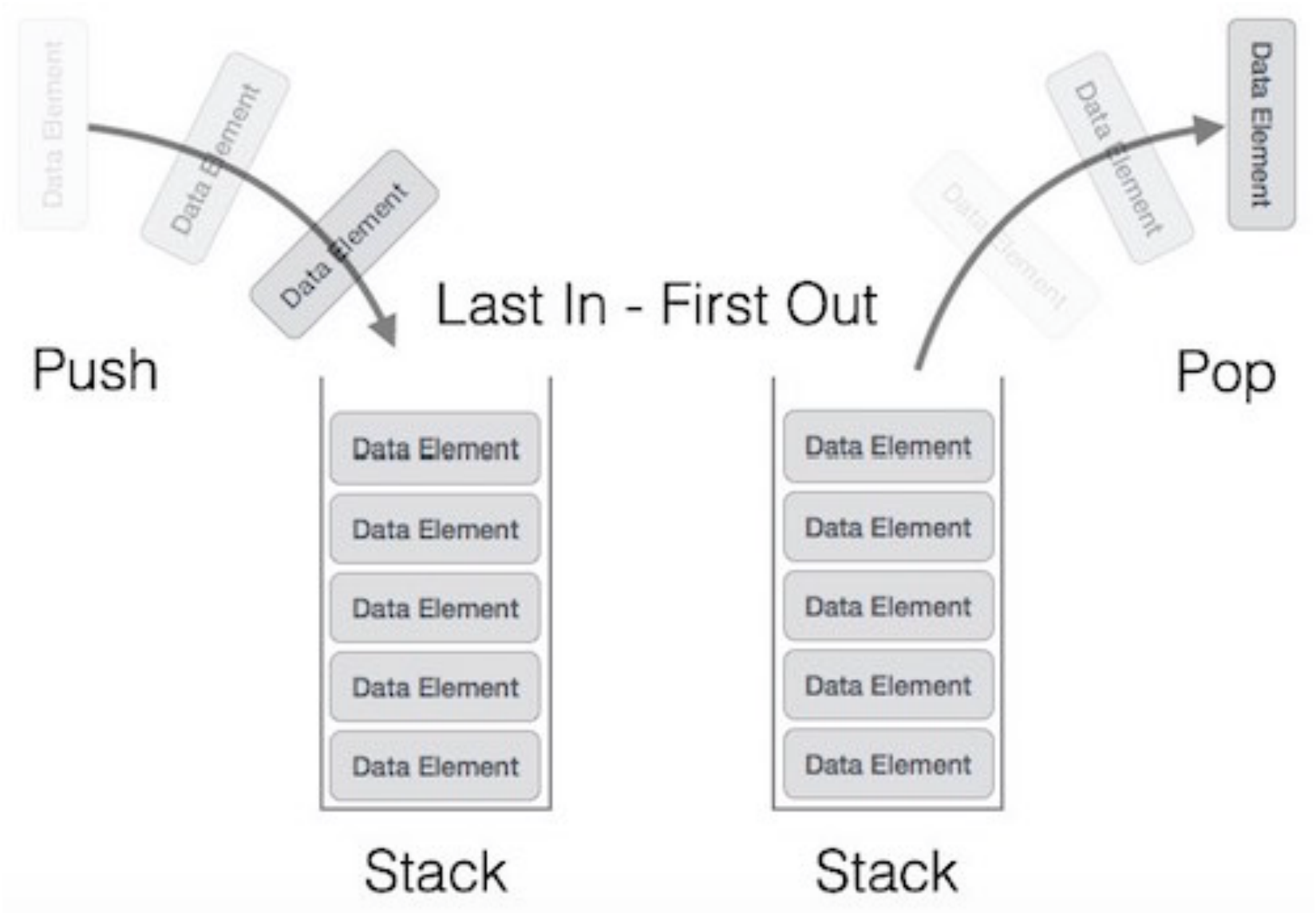
- Triển khai được cấu trúc Stack
- Triển khai được cấu trúc Queue
- Triển khai được cấu trúc Tree
- Phân biệt được các trường hợp sử dụng giữa Stack, Queue và Tree

Stack

Stack

- Giống như một ngăn xếp
- Hoạt động theo nguyên lý Last-In-First-Out
- Có thể dùng Mảng hoặc là một linkedList để lưu trữ các phần tử

Cấu trúc của Stack



Các thao tác cơ bản của Stack

- **push()**: lưu giữ một phần tử trên ngăn xếp
- **pop** (): Xoá một phần tử từ ngăn xếp
- **peek()**: lấy phần tử dữ liệu ở trên cùng của ngăn xếp, mà không xóa phần tử này.
- **isEmpty**(): Kiểm tra rỗng
- **isFull()**: kiểm tra xem ngăn xếp đã đầy hay chưa.

Cài đặt lớp Stack

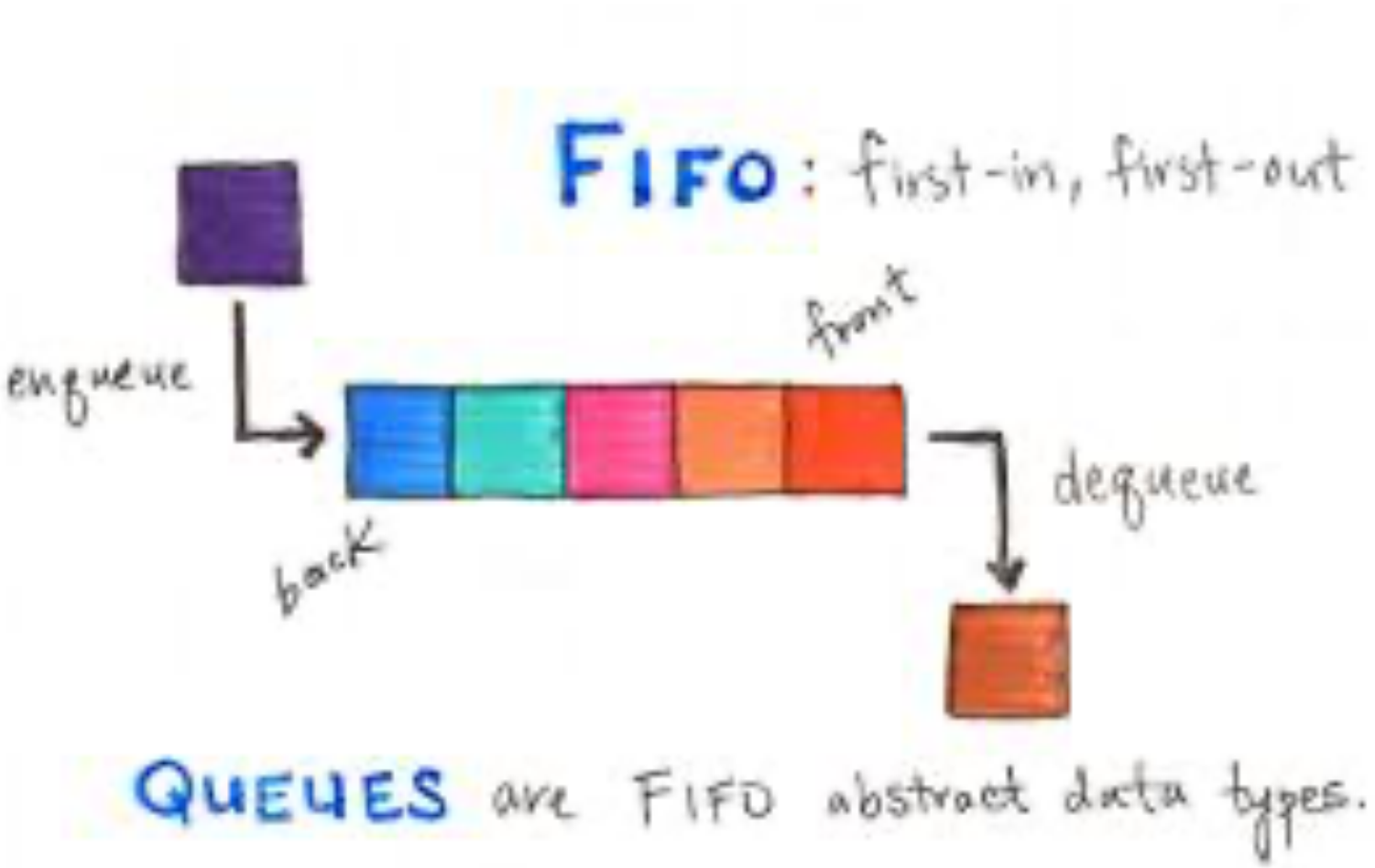
```
class Stack
{
    protected $stack;
    protected $limit;
    public function __construct($limit = 20){...}
    function push($item){...}
    function pop(){...}
    function isEmpty(){...}
    function top(){...}
    function isFull(){...}
}
```

Queue

Queue

- Là một cấu trúc dữ liệu trừu tượng, là một cái gì đó tương tự như hàng đợi trong đời sống hàng ngày (xếp hàng).
- Tuân theo phương pháp First-In-First-Out, tức là dữ liệu được nhập vào đầu tiên sẽ được truy cập đầu tiên.

Cấu trúc của Queue



Các thao tác cơ bản của Queue

- **enqueue()**: thêm (hay lưu trữ) một phần tử vào trong hàng đợi.
- **dequeue()**: xóa một phần tử từ hàng đợi.
- **peek()**: lấy phần tử ở đầu hàng đợi, mà không xóa phần tử này
- **isEmpty**(): Kiểm tra rỗng
- **isFull()**: kiểm tra xem hàng đợi đã đầy hay chưa.

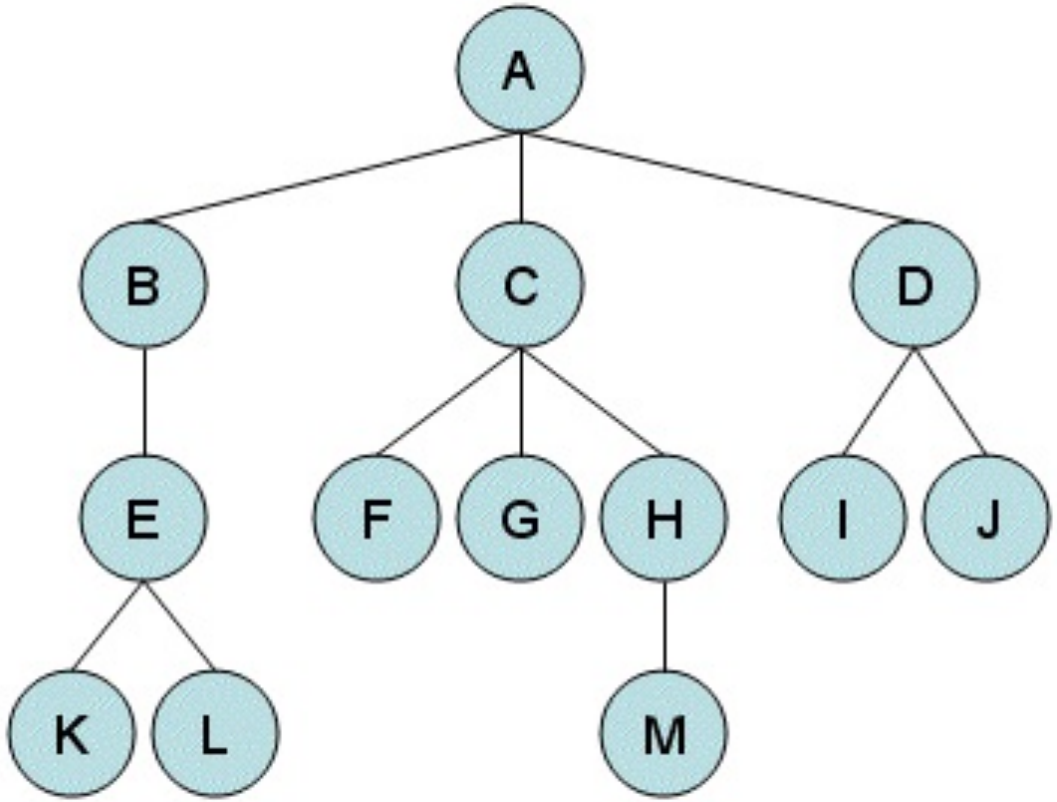
Cài đặt lớp Queue

```
class Queue
{
    protected $limit;
    protected $queue;
    function __construct($limit = 20){...}
    function enqueue($item){...}
    function dequeue(){...}
    function isEmpty(){...}
    function isFull(){...}
}
```

Tree

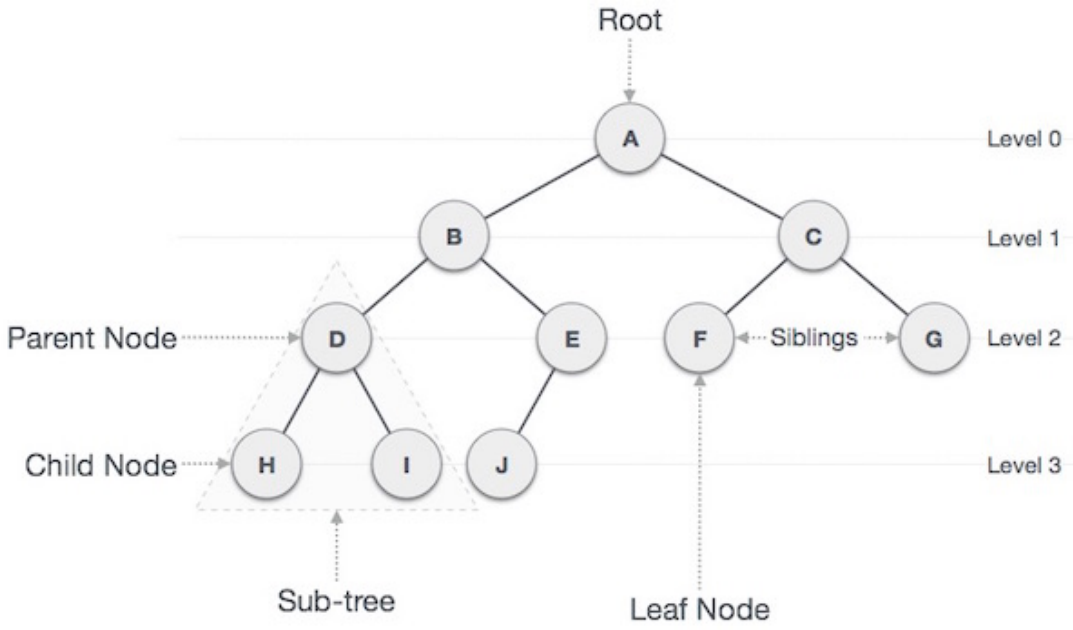
Tree

- Là một cấu trúc dữ liệu được sử dụng rộng rãi gồm một tập hợp các nút (tiếng Anh: *node*) được liên kết với nhau theo quan hệ cha-con.

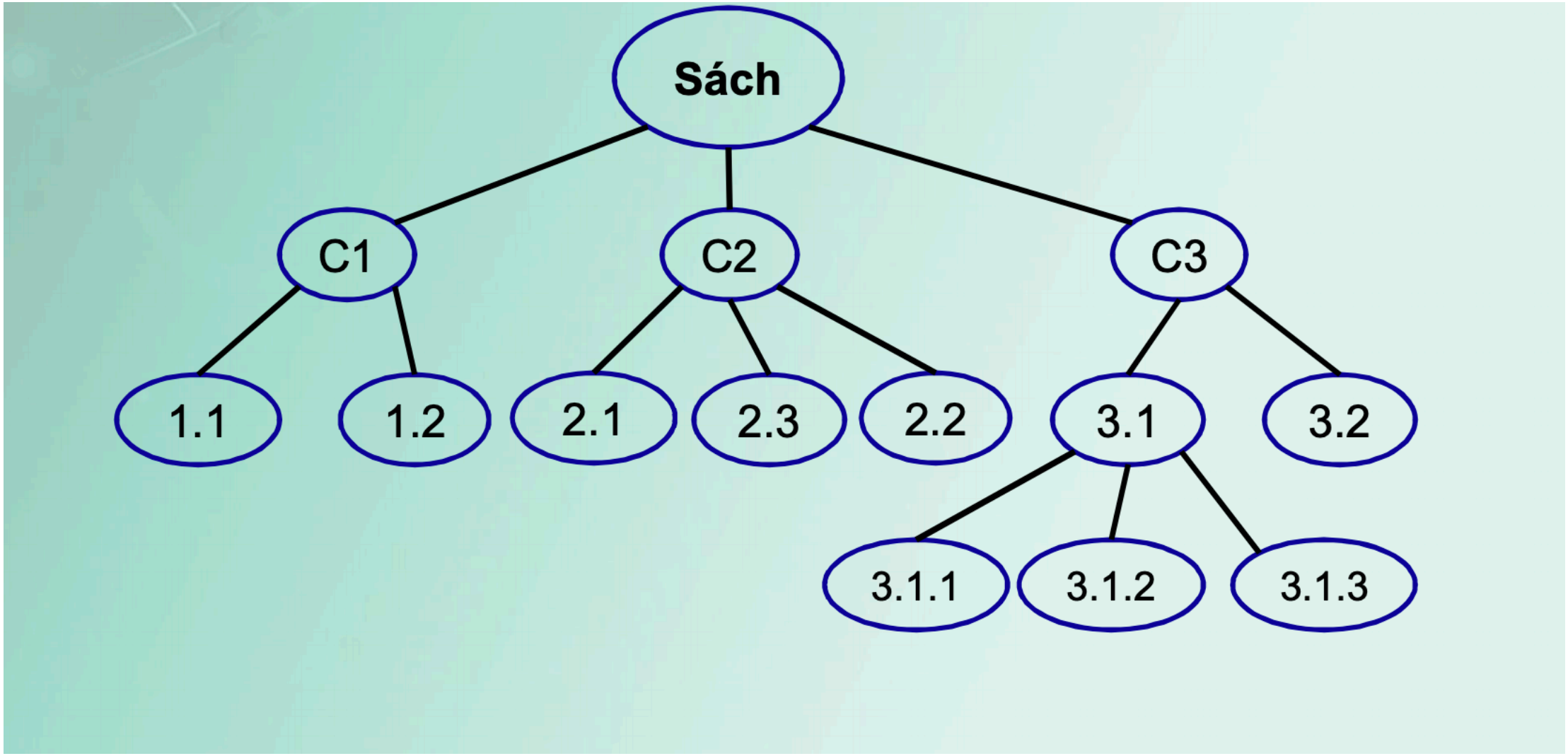


Các khái niệm cơ bản về cây

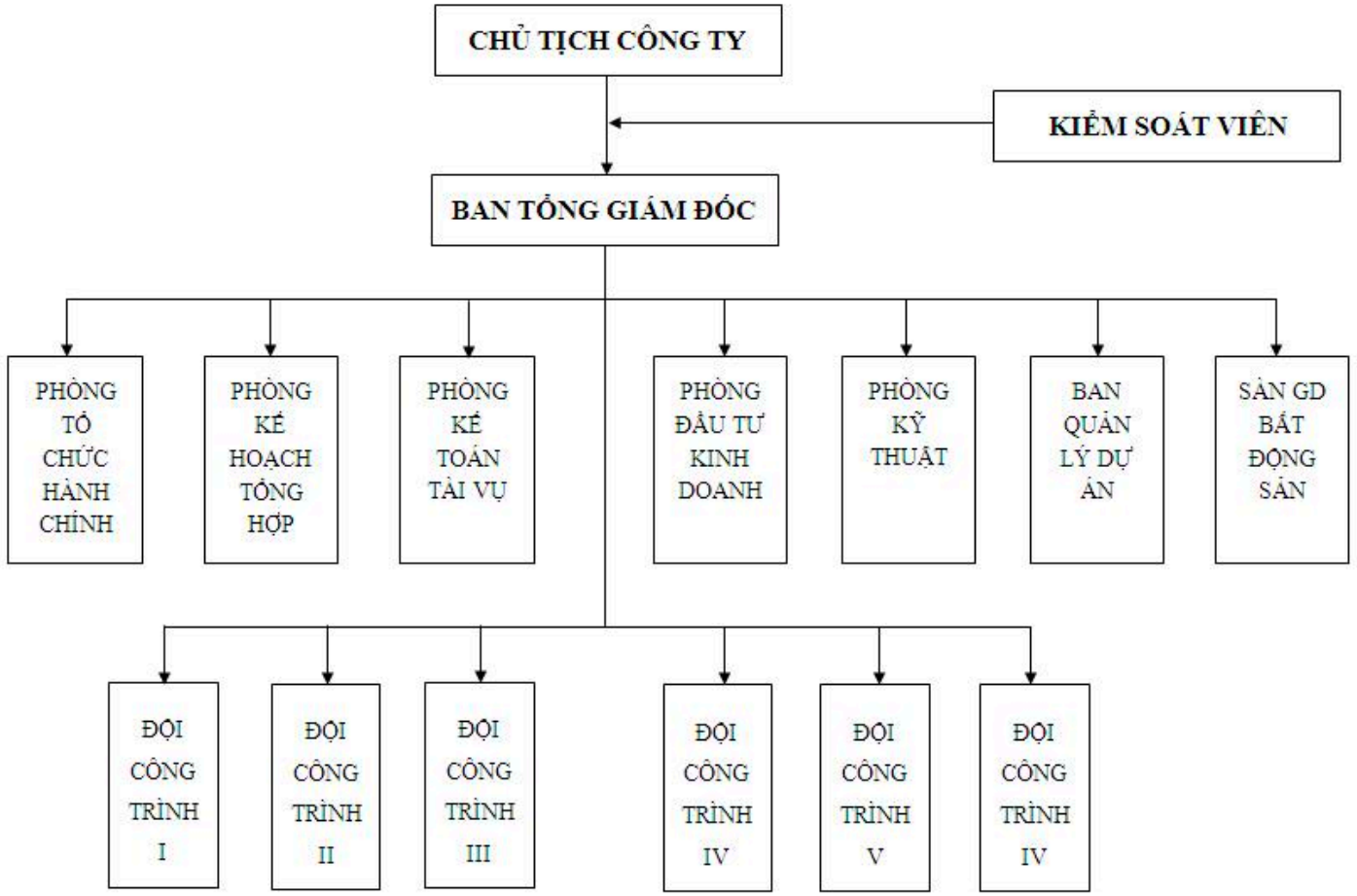
- Nút gốc (Root) – duy nhất
- Nút cha
- Nút con
- Nút lá
- Cây con
- Duyệt
- Bậc
- Khóa (Key)



Ví dụ: Cây mục lục của quyển sách

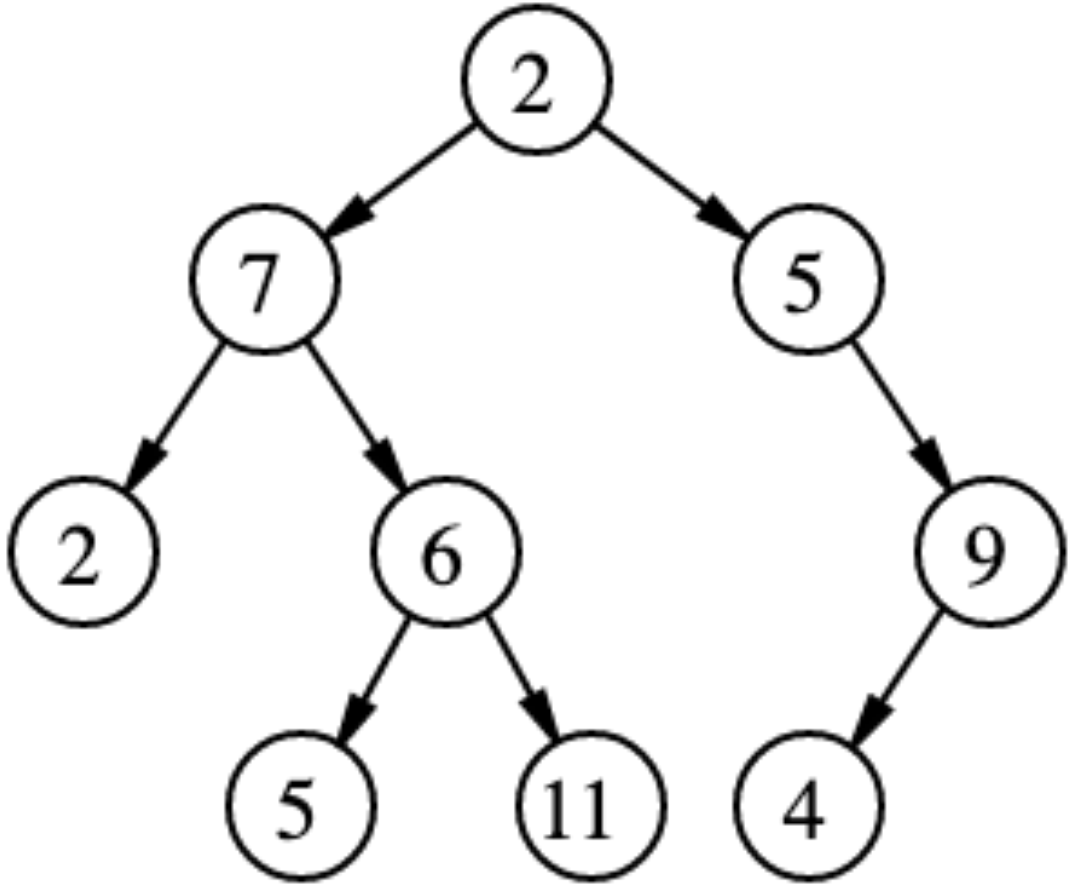


Ví dụ: Cây sơ đồ tổ chức trong công ty



Cây nhị phân - Binay Tree

- Là cây rỗng hoặc là cây mà mỗi node có tối đa 2 node con



Cài đặt cây nhị phân - Binay Tree

- Cài đặt lớp BinaryNode

```php
class BinaryNode
{
    public $data;
    public $left;
    public $right;

    public function __construct($data = null){...}
    public function addChildren(BinaryNode $left,
                                BinaryNode $right){...}
}
```

Cài đặt cây nhị phân - Binay Tree

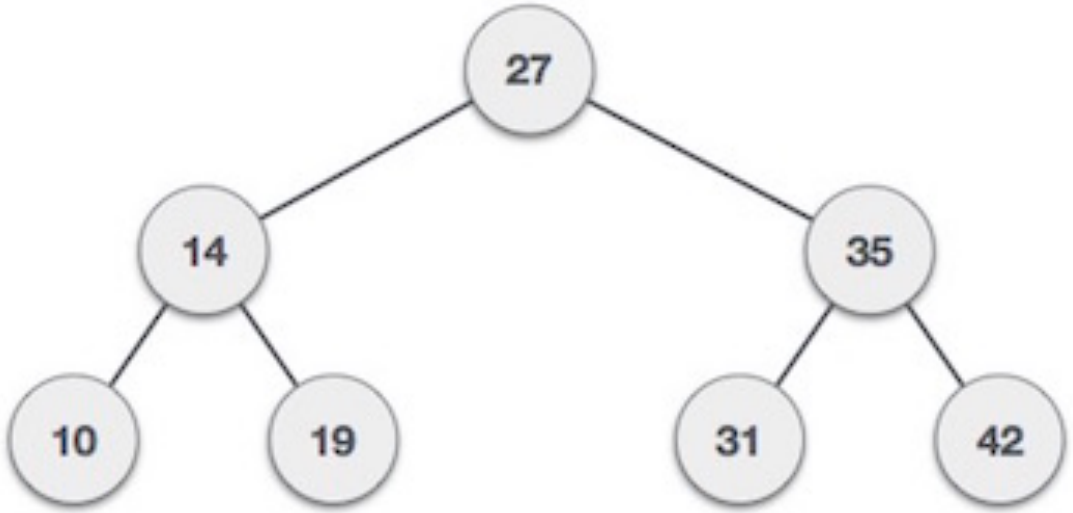
- Cài đặt lớp BinaryTree

```php
class BinaryTree
{
    public $root = null;
    public function __construct(BinaryNode $node){...}
    public function traverse(BinaryNode $node, $level = 0){...}
}
```

Cây nhị phân tìm kiếm - Binay Search Tree

Là một cây nhị phân mà tại mỗi node:

- Phần tử ở node đó lớn hơn các phần tử ở cây con bên trái
- Nhỏ hơn các phần tử ở cây con bên phải



Hoạt động cơ bản trên cây tìm kiếm nhị phân

- **Chèn**: chèn một phần tử vào trong một cây/ tạo một cây.
- **Tìm kiếm**: tìm kiếm một phần tử trong một cây.
- **Duyệt tiền thứ tự**: duyệt một cây theo cách thức duyệt tiền thứ tự
- **Duyệt trung thứ tự**: duyệt một cây theo cách thức duyệt trung thứ tự
- **Duyệt hậu thứ tự**: duyệt một cây theo cách thức duyệt hậu thứ tự

Tóm tắt bài học

- Stack lưu trữ dữ liệu hoạt động theo nguyễn lý Last-In-First-Out
- Có thể sử dụng Mảng hoặc là một linkedList để lưu trữ các phần tử trong stack
- Khác với Stack, Queue hoạt động theo nguyên tắc First-In-First-Out
- Cấu trúc dữ liệu Tree được sử dụng để lưu trữ dữ liệu, một dạng của cấu trúc dữ liệu Tree hay sử dụng đó là Cây nhị phân tìm kiếm.

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***PHP Datastructure***