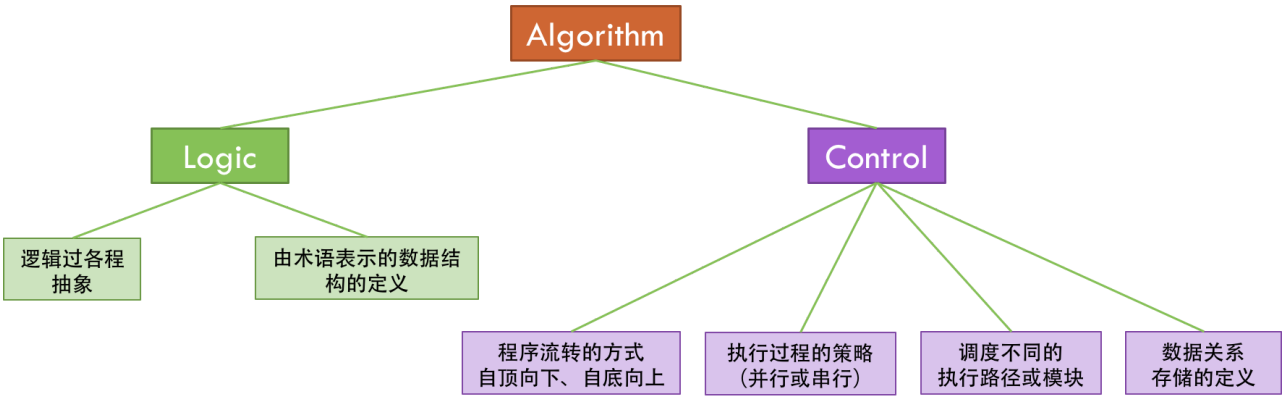
Program = Logic + Control + Data Structure

如果你再仔细地结合我们之前讲的各式各样的编程范式来思考上述这些概念的话，
你是否会觉得，所有的语言或编程范式都在解决上面的这些问题。也就是下面的这几个事。

- Control 是可以标准化的。比如：遍历数据、查找数据、多线程、并发、异步等，都是可以标准化的。
- 因为 Control 需要处理数据，所以标准化 Control，需要标准化 Data Structure，我们可以通过泛型编程来解决这个事。
- 而 Control 还要处理用户的业务逻辑，即 Logic。所以，我们可以通过标准化接口 / 协议来实现，我们的 Control 模式可以适配于任何的 Logic。

上述三点，就是编程范式的本质。
有效地分离 Logic、Control 和 Data 是写出好程序的关键所在!
有效地分离 Logic、Control 和 Data 是写出好程序的关键所在!
有效地分离 Logic、Control 和 Data 是写出好程序的关键所在!



代码复杂度的原因：
业务逻辑的复杂度决定了代码的复杂度；
控制逻辑的复杂度 + 业务逻辑的复杂度 ==> 程序代码的混乱不堪；
绝大多数程序复杂混乱的根本原因：业务逻辑与控制逻辑的耦合。
如何分离 control 和 logic 呢？我们可以使用下面的这些技术来解耦。
State Machine
状态定义
状态变迁
条件状态的 action
DSL – Domain Specific LanguageHTML，SQL，Unix Shell Script，AWK，正则表达式……
编程范式面向对象：委托、策略、桥接、修饰、IoC/DIP、MVC……
函数式编程：修饰、管道、拼装
逻辑推导式编程：Prolog
这就是编程的本质：Logic 部分才是真正有意义的（What）Control 部分只是影响 Logic 部分的效率（How）