面向过程（Procedure Oriented，简称PO）

设计思路：数据结构 + 算法

> - **编程思想：** 以过程为中心，分析出解决问题所需要的功能，按功能划分为若干个基本模块，使用的时依次调

设计优点

> - **效率高：** 善于结合数据结构来开发高效率的程序
> - **流程明确：** 具体步骤清楚，便于节点分析
> - **编程任务明确：** 在开发之前基本考虑了实现方式和最终结果

设计缺点

> - 开发和维护困难，可重用性差、易复用性差、数据安全性差、难以开发大型软件和图形界面的应用软件

面向对象（Object Oriented, 简称OO）

设计思路：由现实世界建立软件模型

> **编程思想：**
>
> - 以事物为中心，把构成问题事务分解成各个对象
> - 建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为

设计优点

> - **封装性：** 将事务高度抽象，便于行为分析、操作和自省
> - **容易扩展：** 代码重用率高，可继承，可覆盖
> - **结构清晰：** 程序便于模块化，结构化，抽象化，更加符合人类的思维方式
> - **实现简单：** 可有效地减少程序的维护工作量，软件开发效率高

设计缺点

> - **复杂性：** 过度的封装导致事务本身的复杂性提高
> - **效率低：** 在面向过程的基础上高度抽象，导致和代码底层的交互机会少，不适合底层开发和游戏甚至多媒体开发

设计概念

> - **类：具有相同属性和行为的一组对象的集合** （数据类型）
> - **对象：类的实例**（数据类型定义的变量名）
> - **封装：**将属性和行为作为一个整体，表现生活中的事物，并将属性和行为加以权限控制
> - **多态：**同样的调用语句有多种不同的表现形态；同样一个函数在不同的子类、父类穿梭的时候表现出不同的形态
> - [x] **属性：描述对象静态特征的数据项** （变量）
> - [x] **行为：描述对象动态特征的操作序列** （行为函数）

C/C++

> - **C语言缺点：**
> - [x] 没有深思熟虑的设计过程
> - [x] 使用时存在很多“灰色地带”
> - [x] 残留量过多低级语言的特征
> - [x] 直接利用指针进行内存操作
> - **C语言优点：**
> - [ ] 执行效率高

> - C++ *以C语言为基础*，且*完全兼容C语言*的特性
> - **C++: C语言 + 面向对象支持 + 函数加强 + 类型加强 + 异常处理**
