# 概论小结

- 软件 = 程序+数据+文档
- 计算机系统：软件与硬件、数据库、人、过程
- 软件危机:现象、原因、办法 (软件工程学)
- 软件工程:开发、运行和维护软件的系统方法
- 软件工程基本原理
- 软件工程3个要素：工具、方法和过程。
- 软件工程四个原则：采用适宜的开发模型，使用恰当的开发方法，提供高质量的工程支持，实施有效的工程管理
- 软件生命周期:定义、开发、运行维护
- 软件生命周期过程:过程模型

# 可行性分析小结

- 可行性研究的工作任务和过程
- 跨职能业务流程图
- 数据流图
- 数据字典

# 需求分析小结

- 怎样将跨职能流程图转化为数据流图
- 数据流图的画法和要求
- 怎样从数据流图得出实体-联系图
- 实体-联系图的含义（实体、属性、关系、主键等）
- 数据规范化原理，为何要规范化?
- 数据字典技术
- 状态转换图
- 功能结构层次图

# 详细设计小结

- 详细设计目标：不是具体地编写程序，而是要设计出程序的“蓝图”，确定怎样具体地实现所要求的系统，详细设计的结果基本上决定了最终的程序代码的质量。
- 详细设计任务：确定每一模块的算法、所使用的数据结构、外部接口和用户界面、测试用例。
- 结构程序设计经典定义：程序的代码仅仅通过顺序、选择和循环这3种基本控制结构进行连接，每个代码块只有一个入口和一个出口，尽可能少用GOTO语句。
- 掌握界面设计的10个基本原则，了解界面设计的通用规范
- 了解主从表设计和代理主键（非业务逻辑主键）

# 实现小结

- 系统实现：程序编码是设计的继续，程序员须熟练掌握并正确地运用程序设计语言的特性，源程序具有良好的结构和良好的程序设计风格
- 编码原则：简单性原则、可读性原则、自顶向下，逐步求精、可调试
- 系统实现的任务：编写源程序、编写文档、单元测试
- 编码风格：程序也是一种供人阅读的文章，编码需要遵循一定的标准规范
- 程序文档化：符号命名（要有实际意义、统一的命名规则）、程序注释（序言性注释、功能性注释）、数据说明（次序规范化、变量有序化、注释复杂数据结构）、语句结构（一行只写一条语句、首先考虑清晰性、直截了当地说明程序用意、尽可能使用库函数、避免不必要的转移（goto）、避免采用过于复杂的条件、减少使用“否定”条件的条件语句、大程序要分块编写、尽量避免使用递归过程）
- 程序效率：与详细设计阶段确定的算法效率有直接关系，程序效率与程序的简单性相关
- 程序效率原则：优秀的算法、简化表达式、少嵌套、避免复杂结构，先使程序正确和清晰，再使程序有效率

# 测试小结

- 测试的目的：尽可能发现并排除软件中潜藏的错误，证明软件是错误的，而不是证明软件是正确的
- 测试准则：尽早测试、追溯到需求、2/8原则、第三方测试、回归测试
- 测试对象：整个生命周期，不仅仅测试程序，包括文档
- 测试步骤：单元测试、集成测试、确认测试（ Alpha开发场所， Beta用户场所）、系统测试（上线前，真实环境测试）
- 测试分类：静态测试（人工和计算机相结合）、动态测试（黑盒测试和白盒测试）
- 白盒测试技术：参考设计文档,程序内部逻辑结构（路径、控制、循环）
- 黑盒测试技术：参考需求规格,重点测试程序功能界面（等价划分、边界值），错误推测（容易发生错误的模块、功能复杂的模块、测试时已发生很多错误的模块、团队中能力偏弱的人（责任心不强的人））

# 维护小结

- 1、维护是软件生命周期的最后一个阶段，也是持续时间最长、代价最大的一个阶段。**软件工程学的主要目的就是提高软件的可维护性，降低维护的代价**。
- 2、软件维护通常包括4类活动：**改正性维护、适应性维护、完善性维护、预防性维护**。
- 3、软件的**可理解性、可测试性、可修改性、可移植性和可重用性**，是决定软件可维护性的基本因素。
- 4、在软件生命周期的**每个阶段都必须充分考虑维护问题**，并且为软件维护预做准备。
- 5、**文档是影响软件可维护性的决定因素**。
- 6、在条件具备时应该**主动地进行预防性维护**。
- 7、**预防性维护实质上是软件再工程**。典型的软件再工程过程模型定义了库存目录分析、文档重构、逆向工程、代码重构、数据重构和正向工程6类活动。