18-20

- **chp18 表驱动法**
  - 18.1 表驱动法使用总则
    - 确定如何从表中查询数据
    - 1 直接访问
    - 2 索引访问：
    - 3 阶梯访问
  - 18.2 直接访问表
    - 下标数组获取每月天数
    - 状态机，拆分成小块逻辑，配置化选择执行多个
  - 18.3 索引访问表
    - 数据库字段索引
  - 18.4 阶梯访问表
  - 18.5 表查询的其他示例
- **chp19 一般控制问题**
  - 19.1 布尔表达式
    - 简化复杂的表达式
      - 拆分复杂判断，引入中间布尔变量
      - 拆分成方法，隐藏细节
      - 用表驱动法管理变量
    - 编写肯定形式的表达式
      - !statusOk -> statusError
    - 善用括号，使表达更清楚
    - 理解布尔表达式如何求值
  - 19.2 复合语句块
  - 19.3 空语句
  - 19.4 驯服危险的深层嵌套
  - 19.5 编程基础：结构化编程
    - 三个组成部分
      - 顺序

- 选择
- 迭代
- 任何一种控制流都可以由顺序、选择、迭代三种结构生成。

- 19.6 控制结构和复杂度
  - 复杂度的重要性
  - 降低复杂度的一般原则
    - 如何衡量：决策点的个数
    - 大于10 的复杂度：拆分子程序
    - 子主题 3

chp20 软件质量概述

- 20.1 软件质量的特性
  - 1 可维护性： 是否能够很容易的进行修改，改变或增加功能，性能优化，修复缺陷
  - 2 灵活性：从特定用途、特定环境迁移到别的场景的能力
  - 3 可移植性：
  - 4 可重用性：应用到其他系统的难易程度
  - 5 可读性：阅读理解代码的难易程度，尤其是细节上
  - 6 可测试性：何种程度上进行系统验证、测试
  - 7 可理解性：系统组织结构上理解系统的难易程度
- 20.2 改善软件质量的技术
  - 汇总
    - 软件质量目标
    - 明确定义质量保证工作
    - 测试策略
    - 软件工程指南
    - 非正式技术复查
    - 外部审查
  - 开发过程
    - 对变更进行控制的过程
      - 代码组织兼容变化的能力

- 结果的量化
- 制作原型
  - 设计更完善
  - 更容易维护
- 设置目标
  - 善用成就激励
- 20.3 不同质量保障技术的相对性能
  - 缺陷检测率
  - 找出缺陷的成本
  - 修正缺陷的成本
  - 推荐套路
    - 对所有的需求、架构以及系统关键部分的设计进行正式检查
    - 建模或创建原型
    - 代码阅读或者检查
    - 执行测试
- 20.4 什么时候进行质量保障工作
  - 需求早期就要引入
- 20.5 软件质量的普遍原理

- **选择一种好的方案而又避免发生灾难，不要试图去寻找最佳方案**