总 结

总结主要是两部分

1. 贯穿本课程（教科书）整体的具有普遍意义的重要原理，编程和软件开发的重要原则
   - 抽象，程序的复杂性控制
   - 统一设计的意义和价值
2. 本课程涉及的主要技术内容
   - Scheme 语言和基本程序设计
   - 一些具有通用性的高级技术

最后是复习和考试的一些情况说明

软件的复杂性和控制

- 软件是人类开发的最复杂的系统
  - 指令到整个软件/指令执行时间到软件运行时间/...
- 系统本身的复杂性
  - 规模庞大
  - 参与工作的人众多，主要靠手工和智慧，很难管理
  - 静态描述与动态行为之间的关系，很难把握
- 环境和需求的动态演化带来的复杂性
  - 复杂系统难完全理解，开发过程中新认识带来系统构造的变化
  - 错误难以避免，修改的影响难以把握
  - 需求不断变化，不断出现维护和提升功能的新需要
- 系统的良好设计，对于支持修改、变动、升级等活动至关重要
  - 本书主要不是讨论巧妙的算法，而是讨论系统的组织和设计

系统设计和抽象

- 控制和管理系统（程序）的复杂性，主要靠两个相关的指导思想
  - 分解和隔离：将复杂系统分解为一些相互隔离又相互联系的部分
  - 抽象：从每个部分看，与它有联系的各部分都是某种抽象。每个抽象有清晰的接口，明确的功能，可以方便地使用
- SICP 的基本观点：通过良好设计的抽象，实现系统中
  - 各部分之间的良好隔离
  - 每个部分的清晰接口
  - 实现细节信息的隐藏
- SICP 提出了三层抽象的概念（全书围绕着这几个核心概念）
  - 过程抽象（及其实现的技术和问题）
  - 数据抽象（及其实现的技术和问题）
  - 语言抽象（及其实现的技术和问题）

抽象

- 过程抽象
  - 抽象技术：定义过程（函数）
  - 封装：有关计算进程的实现细节
  - 接口：过程的使用方式（参数，返回值等）
  - 原则：一切可以定义为过程的计算片段都应该定义为过程
- 数据抽象
  - 抽象技术：定义一组接口过程，或定义接收一组消息的过程
  - 封装：对象的实现细节，对象的状态
  - 接口：一组接口过程或者一组消息
  - 系统的状态信息应分属于一批接口清晰的对象
    - 首先设计好数据抽象的接口，隔离具体实现
    - 尽可能推迟实现细节的设计决策，直至有了清晰的认识

抽象

- SICP 中有大量数据抽象设计的实例。设计中的活动：
  - 遇到可能需要保存状态和信息，需要表示某种“事物”，立刻想到应该定义一种数据抽象来表示它
  - 设计该数据抽象的接口过程（构造，访问，变动状态的操作）
  - 研究有关接口是否满足使用的需要，并做必要的修改
  - 适当的时候再研究（再完成）相关的实现
- 语言抽象
  - 抽象技术：定义和实现专门用于解决一类应用问题的语言
  - 封装：某个应用领域里的典型构造和处理过程
  - 接口：一组语言原语、构造机制和抽象机制，支持应用领域基本概念的构造、组合和抽象，以方便具体问题的描述
  - 原则和问题：要慎重启动，最终需要考虑和完成语言的实现

统一的设计

- 书中许多实例显示了统一设计的美妙和威力，值得学习
- 在一个软件或者一个软件部件里
  - 应该有一个清晰的设计想法和一套清晰且统一的实现方法
  - 在设计和实现两个方面都应该有统一性
- 在 SICP 书中可以看到许多例子
  - 书中所有复杂的例子都贯彻了这一原则
  - 希望大家自己重新看看这些例子（学期结束以后）
  - 其中许多想法都值得借鉴
- 书中的许多做法有参考价值。建议大家（有兴趣和时间）想想如何在你熟悉的其他语言或环境里实现 SICP 的实例
  - 应该如何设计相关的过程、数据抽象和语言
  - 分析采用不同语言的优势和劣势，可能怎样缓解语言劣势的影响

课程总结

下面简单总结本课程和教科书的主要内容

主要分为三部分

1. **Scheme** 语言的基本概念
   - 基本编程元素
   - 组合机制
   - 抽象机制
2. **Scheme** 程序设计
3. 具有普遍意义的原理

Scheme 语言

- 基本结构：数, 符号, 字符串, 组合表达式, 过程定义
- 基本过程和基本谓词：**cons, car, cdr (包括 cadr 等), +, -, *, /, quotient, remainder, abs, min, max, nil 等； null?, eq?, equal?, pair?, =, >, <, >=, <=, even?, odd?, number?, zero?, symbol? 等**
- 特殊形式：**if, cond, begin, and, or, not, quote**
- 表操作：**list, append, length, map, assoc, memq, member**
- 高阶过程：**map, ...**
- 过程定义和局部状态等：**define, lambda, let, eval, apply**
- 修改状态：**set!, set-car!, set-cdr!**
- 基本流操作：**cons-stream, stream-car, stream-cdr, stream-null?, the-empty-stream, delay, force**
- 输入输出：**read, display, newline**（考试中不要求做输入输出）
- 求值模型：代换模型，环境模型

Scheme 程序设计

- 过程产生的典型计算：线性迭代，线性递归，树形递归
- 基本过程的应用和复合表达式（直接描述的计算）
  - 表达式的基本求值过程
  - 应用序和正则序求值，意义，差异，什么时候结果不同
- 过程定义（控制抽象，基本抽象机制）
  - 基本过程定义
  - 递归程序设计，基本技术
  - 实现线性迭代，线性递归，树形递归
  - 局部过程定义，局部过程中的非局部名字的意义（作用域规则）
  - 高阶过程：以过程作为参数或返回值
  - 用 **lambda** 描述过程
  - 用 **let** 表达式定义局部变量

Scheme 程序设计

- 数据抽象（基本概念，编程原则，得与失）
  - 直接定义：数据表示和接口过程
  - 采用带标志数据，用于区分不同类型的对象，创建通用过程
- 结构化数据的基本操作技术
  - **cons/car/cdr** 和 **list** 等（构造和解析）
  - 表的基本处理模式：顺序递归，结果的累积或构造
  - 将表看作树，相应的处理：按层次递归
  - 高阶表操作函数，**map** 操作模式（逐个应用，得到结果的表），其他操作模式
  - 带标志数据（用首元素作为标志，区分不同类型的数据）
  - 带标志数据的剖析、成分选取和构造
  - 复杂结构的构造

Scheme 程序设计

- 基于对象和状态的程序设计
  - 局部状态（局部变量）和状态变动（**mutation**），操作 **set!**
  - 表结构的变动，操作 **set-car!** 和 **set-cdr!**
- 消息传递风格的程序设计
  - 基本技术：生成带有局部状态的过程
  - 用一个基本过程作为消息分发接口
  - 可以定义一组内部过程实现对局部状态的操作（包括修改状态）
- 语言处理器（解释器、分析器、求值器等等）
  - 基本框架是按结构分发（**eval**），调用适当的处理过程
  - 处理抽象结构的过程（**apply**），根据语言的抽象机制设计
  - 可能需要建立环境（全局和局部状态），支持多层抽象的执行
  - 不同语言的实现有许多共性

有普遍意义的论题

- 程序设计语言的基本结构：基本功能，组合机制和抽象机制
- 基本程序组织：过程抽象（控制抽象）和数据抽象
- 无状态程序设计（函数式程序设计）和基于状态的程序设计（基于局部变量和赋值，基于状态变化）：
  - 两种编程范型各自的得失，优势与劣势
- 求值的环境模型。适用于一般程序，具有普遍意义
  - 局部环境和外围环境（非局部变量的处理，作用域规则）
  - 环境的扩充和恢复，过程调用的局部环境
  - 过程对象（带有环境），过程调用时的环境转换（静态作用域）
- 赋值与时间、并发的关系（考试不会涉及）
  - 并发带来的问题：非确定性，序列化，互斥，死锁等
- 一般性问题：计算，图灵机和通用图灵机

有普遍意义的论题

- 基于逻辑和关系的程序设计
  - 逻辑程序设计，约束程序设计
  - 非确定性的计算，多结果表达式
  - 无方向的计算（约束传递，逻辑关系的维持）
- 无穷数据结构和惰性计算（懒求值，消极求值）
  - 应用序和正则序求值，惰性计算与正则序求值的关系
  - 带记忆的惰性计算
- 语言处理过程中的静态处理阶段（编译时/翻译时/分析时/解析时）和动态处理阶段（运行时）。不同处理方式
  - 元循环求值器和分析求值器的对比
  - 解释和编译（翻译的深度）。本书最后的编译器，静态处理阶段的结果是与 **Scheme** 程序对应的寄存器机器语言程序（这次课没讲）

考试问题

- 时间和地点：**6** 月 **18** 日（星期三）晚 **6** 点半到 **8** 点半，理教 **106**
- 考试方式：闭卷笔试，在课程成绩中占 **50** 分
- 答疑：**17**日（周二）下午 **2-5** 点，**18**日（周三）上午 **9-11**点，理科楼 **1479**。其他时间可到我办公室（理科**1**号楼 **1480**，只要我在）
- 题目形式：
  - 与基本概念和基本模型有关的简要回答题（不需要背书）
  - 课程中反复讨论的有关计算、程序和程序组织的问题
  - 编程题。主要是各种过程的定义和使用，不涉及输入输出
    - 要求：意义正确，结构和格式清晰。可根据需要加注释
    - 简单计算，过程定义，递归过程，表处理，结构数据的分解和构造，基于变动状态的程序设计，基于消息传递的程序设计等
    - 过程都很短，只涉及前面提到的基本过程和结构。如用到不常用的过程会有说明