程序设计基础导引

计算机无处不在

- 计算机很强大
  - 计算。科学计算等
  - 管理。信息采集和管理。如，信息管理系统，电子商务
  - 控制。控制实体。
  - 可以被数字化，信息化的事物。多媒体；娱乐；
- 个人家庭
- 公共场所
- 工作场所

为什么计算机能起作用？

- 计算机的逻辑结构（图灵机模型）
  - 指令系统
  - 数据/信息机制
  - 给定输入，按照指令执行，输出
- 计算机的物理结构（冯诺依曼模型）
  - 中央处理器（CPU，为计算单元）
  - 随机存储器（RAM，为数据存储单元）
  - 外存（外部存储单元）
  - 输入设备和输出设备
  - 二进制表示

计算机科学，逻辑学和数学保证

- 人（生物进化）
  - 大脑
  - 记忆
  - 感官接受输入，肌肉系统等输出
- 程序
  - 指令序列
  - 数据
  - 程序入口出发，按照代码顺序执行，对数据进行处理，最后停止

什么是程序设计

- 程序设计，也称计算机编程，将人的想法、思维过程转换成计算机要执行的动作
- 程序设计（编程）语言，在程序员和计算机之的桥梁
  - 将人的语言转换成计算机“懂”的语言
- 程序
  - 用编程语言编写的，完成特点功能的语句集合

编程意味着什么？

程序设计（编程）是用来解决问题的。是程序员意图的实现

- 首先，问题可以用计算机来解决
  - 可以转换到计算机里（建模）；模型可以被计算
- 其次，建模（问题抽象）
  - 将具体问题抽象，用数据（结构）表示事物，用算法表示求解过程
  - 建模方式，可以是自顶向下，也可以自底向上
- 最后，求解（编程实现）
  - 用变量等表示数据；
  - 用语句，函数等（代码/代码模块）表示具体的计算过程

如果没有正确算法的程序，计算机就不会得到预期的结果

编程意味着什么？

- 我们讨论编程，我们是在说
  - 用数据结构（有结构的数据/集）来表示问题的模型
  - 用算法来表示问题的求解
  - 数据结构和算法往往合在一起考虑

- 科学计算
  - 首先得知道怎么算
  - 用变量等存储数值
  - 用数学相关的指令计算数学表达式
  - 用顺序，条件，循环语句控制计算步骤
- 信息管理
  - 首先，事务可以被信息化；
  - 将事务抽象；用变量（结构体，数组）表示各种信息
  - 编写代码模块实现信息录入，信息检索功能

- 游戏
  - 抽象
    - 僵尸，植物等用变量（对象）表示；
    - 判断子弹是否击中僵尸等，用函数表示，判断碰撞
    - 生命值用数值表示，小于0，死了，从数据集中删去
  - 实现
    - 实现数据和函数等
    - 补充其它。如画出僵尸等，以及完整游戏流程的控制
- 音乐播放
  - 模型抽象
    - 将音乐数据变成音频的二进制序列。驱动外部设备。
  - 实现
    - 播放：将音频数据发送到播放设备，进行播放
    - 音乐数据管理：文件或者数据库，实现增删改查

开始编程

- Hello world开始
  - 用字符串”Hello world”表示数据
  - printf(“Hello world”);表示操作/计算
- 更难点，数学运算
- 圆面积
  - 半径用小数表示，double r=1.0;
  - 计算用数学运算，以及赋值运算
    - double area=3.1415926*r*r;
  - 输出
- 为了控制计算流程，引入条件语句，if
- 开始累加
  - 使用计算技巧，累加器变量int sum=0;
  - 循环控制语句：for(int i=0; i<100;i++)sum+=I;

当数据变复杂

- 数据规模变大
  - 命名和读写（访问）变得不容易了；必须数据保存
  - 数组
  - 文件存储
- 成分复杂
  - 表示的属性存在关联
    - 结构体
  - 数据和数据之间存在逻辑关联
    - 线性表，树，图等数据结构
- 总之，数据复杂，规模变大之后，带来一系列问题。

当程序规模变复杂

可复用；模块化；不易出错，容易发现和找到问题；容易构建结构良好的复杂系统

- 首先，发展出函数
  - 逻辑上的独立单元
  - 可复用性
- （C++）其次，发展模板
  - 算法对不同数据结构的复用
- 更多的程序设计思想
  - 小到：变量、函数命名方法
  - 大到：模块化思想，面向对象思想等

- 软件设计模式：某些经典任务，提供分析清晰的解决方式
- 软件工程：多个模块、软件，协同完成复杂任务

总之，程序设计需要设计

编写程序的时候，不能只要结果对就行，需要进行仔细的“设计”

- 首先设计好“数据结构+算法”，正确性是第一位的！
  - 正确
  - 通过测试

“好”的程序应该什么样的？

- **可读性**
  - 开发者，合作者都需要读懂
  - 编程风格（代码规范），软件结构，逻辑设计（逻辑规范），都应具备良好的可读性
  - 标准：形式到逻辑保持一致，代码自描述，只需要很少解释和文档
- **可维护性**
  - **大部分代码需要长期维护**
  - 纠正性维护、适应性维护、完善性维护、预防性维护
  - 原则
    - 坚持简单的原则，最有助于提高可维护性
    - 可维护性应在项目开发的一开始就加以考虑
    - 一个软件系统越遵守原则，可维护性越高
- **简洁性**
  - 优雅的代码：形式到逻辑的简洁
  - KISS原则(keep it simple and stupid)
- **效率性**
  - 运行效率：时间复杂度和空间复杂度
  - 实现效率：复用已有模块；模块能复用
- **健壮(鲁棒)性**
  - **代码容错**：程序在面对无效的输入或者是在某些外在压力的情况下，系统能够正确解决问题
- 模块化与设计模式等等编程方法

问题求解的一般方法

一般的方法：感知->认知->决策->行动

- 定义问题（理解问题）
  - 抽象
    - 剔除那些与问题无关的信息
    - 提取和问题相关的属性
  - 判断方法
    - 转述给他人
- 分析问题
  - 按照逻辑，拆分问题
  - 常用两种分解策略：自顶向下，自底向上
- 设计算法/方案
  - 权衡各种解决方案
  - 核心：方案应分解到足够细粒度
    - 可以对应到代码级别
- 实现
  - 把描述的策略转换代码

一般的程序设计

- 数据类型
  - 字符串（也称为文本类型）
  - 数值型，整数小数等
  - 布尔型，表示真假
  - 结构体，复合结构
  - 数组
- 变量或者对象的概念
  - 承载数据的实体
- 语句
  - 顺序
  - 条件。根据条件，选择执行的语句
  - 循环。满足条件下，反复执行
- 条件语句与循环语句中的条件
  - 用布尔，或者逻辑表达式
- 函数
  - 最小的具有完整逻辑的模块
  - 是模块的最小单位
- 类

集成开发环境（IDE）

- **熟练使用工具，**熟练使用快捷键
- 现代的IDE
  - 具备语法高亮显示功能，能在编写代码时为你做出指导。语法高亮显示功能会更改语句中字符的颜色，方便迅速识别字符的身份，比如，是函数，还是字符串或是其他
  - 自动补全，能帮助你完成语句书写，还不会出现错误
  - 内置的参考资料
- IDE里调试

调试

- 调试，指仔细地检查编程过程、算法、流程图，排除并修复致使程序无法正常运行的问题或错误。
  - 语法类bug是指我们在表达、书写方式方面的错误
  - 逻辑类bug则是指指令执行方式方面的错误以及算法未能按设计进行的问题

其它

- 代码的注释
- 软件文档

- 网络资源搜索
  - Github
  - Stack overflow
  - google

- 使用github
  - 代码管理
  - 协同开发

C++程序设计的要求

- 加深C
  - 加深对程序设计的理解
  - 加深对计算机系统的理解，尤其计算模型，内存机制等部分
  - 从模块化开始的程序设计方法
- 面向对象技术，深入理解其实质
  - 封装
  - 继承
  - 多态
- 模板，理解泛型技术
- 掌握诸如 const，引用，重载等语言机制，理解其用意所在