计算机科学的关键: 控制复杂度(Controlling Conplexity).

计算机科学处理的是: 理想化组件(Idealized Components), 忽略现实的约束.

控制复杂度:

黑盒抽象(Black-Box Abstract), 盒子内部是什么样子不重要,通过此方法掩藏细节, 控制复杂度, 让自己有精力干更大的事情.

元方法(Meta Method): 先对基本对象, 再使用基本过程和基础数据组合起来构建更复杂的东西, 还有组成方法, 抽象方法,再封装为黑盒, 再用黑盒组成更复杂的东西. 是传承.

约定接口(Conventional Interfaces): 实现的方法是按照约定实现相应接口, 并以此将各部分组装起来, 保证新加功能但是不会把系统搞得乱七八糟.

通用操作, 面向对象编程, 关于聚集(Aggergates)的操作, ”流”.

元语言抽象(Metalinguistic Abstraction): 通过构建新的语言隐藏一些复杂度. 解释的过程. 应用-求值循环(Apply-Eva loop)l. 构建Lisp.

学习Lisp:

Primitive Elements(基本元素): Symbols, 过程(Operators: (+-\*/%&|!^), User Defineds), 数据(Numbers: Integer, Double, Boolean, Char, ...), 复合数据和抽象数据(如何?)

Means of Comboation(组合方法): 前缀表示法

Means of Abstration(抽象方法): 语法树

代码示例:

|  |
| --- |
| 编码要求:  对齐参数:  (+ 3  (\* 5  (+ 1 1)))  定义:  (DEFINE INC (X + 1))  (DEFINE SQUARE\_0 (\* X X))  (DEFINE (SQUARE\_1 x)  (\* x x)  ) #最佳定义方式, 其实跟上边没区别, 是个语法糖  # INC定义符号INC, 实现令X自增1的过程  # SQUARE\_0定义符号SQUARE\_0, 实现求X平方的过程  # SQUARE\_1 定义过程SQUARE\_1, 含有一个参数名为X, 实现求X平方的过程  # 块封装: 令过程的中间变量们都作为过程的内部变量  # 关键点: 令语言用户无法分辨过程是内建过程或用户定义过程  (INC 5)  (SQUARE\_0 2)  (SQUARE\_1 3)  SQUARE  output:  6  4  9  [COMPOUND-PROCEDURE SQUARE]  # 布尔运算: 谓词或条件, 左右值, 算符: <>=, 返回: true or false  # 分支判断, 依据布尔运算选择:  (COND ((< X 0)  (TO DO 1)  ELSE  (TO DO 2))  )  (if (> X 0)  (TO DO 3)  )  # 至此到此我已经学会了这个语言 |

计算过程: 操作“数据”，被程序指挥.

编程: 使用代码表达自己的思维.