抽象方法学:用简易的数据构造复杂的系统,层次化系统,构造屏障分离底层,中层高层...

分数运算的实现,但是系统没有提供有理数,没法表达浮点数,怎么解决?

用一种强大的设计策略,即愿望思维ＷＩＳＨＦＵＬ　ＴＨＩＮＧＫＩＮＧ，想象有，而不是拘泥于没有的现实．

|  |
| --- |
| (define (MAKE\_RAT N D))  # RETURNS CLOUD, 创造了一种新的数据结构, 称为”构造函数”  # 假设想表达别的东西, 比如加和乘, 就定义新的过程用来求  # 封装数据, 绑定分子和分母在一起  (NUMBER N (......))  (DENOM D (......))  # 抽数据结构中取数据的过程, 称为”选择函数”  (DEFINE (+RAT X Y))  (........)  )  (DEFINE (\*RAT X Y)  (........)  )  # 定义RAT的Operator |

希望设计语言能够表达脑中的概念,隐藏不必要的信息

把有理数计算分为两部分,向乔治签订契约,乔治提供了把分子分母封装起来放到”云彩”里,需要封装的”胶水”,Lisp提供这种胶水,称为”表结构”,LIST STRUCTURE.

List的基本过程CONS, 返回序对,

|  |
| --- |
| (CONS X Y)  # 序对,X是首部分,Y是尾部分  # 返回一个指向一个盒子的指针,盒子里按首尾装着XY这个序对  (CAR 序对)  (CDR 序对)  # 分别返回X和Y  (DEFINE A (MAKE-RAT 1 2))  (DEFINE B (MAKE-RAT 1 4))  (DEFINE ANS (+RAT A B))  (NUMBER ANS) # CONSOLE OUTPUT 6  (DENOM ANS) # 8  # 遵守了加法规则,但是没有化简  # 化简: 求最大公约数,再取出分子分母,除以公约数,再存入  # 算完后就化简或取出数据时才化简 |

数据抽象: 通过假定的构造函数和选择函数把数据对象与它的表示分割开来的编程方法学.

构造函数和选择函数,过程,定义的符号是抽象层,位于Lisp基本过程序对与分数RAT的算符之间,这是”分离”方法学:分离对象的使用方法,把数据对象的使用和表示分离开.意图:概念和实体分离,让用户可以用其他方法表示数据,比如”分数的化简用哪种表示,化简方法需要按照实际情况做出选择”

把对象规约到最低阶.

在编码前不能把系统设计的完美.

用数据抽象来构建更复杂的东西来控制复杂度.

|  |
| --- |
| (DEFINE (MAKE-VECTOR X Y) (CONS X Y))  # 构造一个二维坐标的构造函数  # 用两个坐标构成向量  # 对象量的操作  # 再用向量构成线段  # 各个数据由抽象层分离开 |

闭包(closure): 闭包性质的组成方法,当用一个东西组成一个东西的时候,也可以用这个东西组成另一个东西.

有了层次系统后如果不使用数据抽象,系统复杂度会有失控的隐患.

用了数据抽象后也要考虑可扩展性和可修改性,要为它的过程命名.

用抽象数据定义:数据的表示,数据的操作(比如算式系统).

契约.使用.忽略抽象层之下.

CONS,序对的实现

|  |
| --- |
| function CONS(X, Y) {  var X, Y;  ......  }  function CONS\_PICK(pick) {  var X, Y;  switch (pick) {  case 0:  return X;  case 1:  return Y;  }  }  function CAR() {  return CONS\_PICK(1);  }  function CDR() {  return CONS\_PICK(1);  } |

建立数据抽象不需要数据本身, 数据和过程的边界要变得越来越模糊.

过程不仅仅是动作的集合,过程是概念实体,是对象,过程并不仅仅是做某件事行为,而是一个真实存在的对象.对象划分领地,互不干扰.