

CSSE 304 Day 14 Summary

1. Some more define-syntax examples: `++-post` `my-and` `for` (live coding)
2. What do we mean by *data?* *information?*
3. What are the three main ingredients of an abstract datatype?
 - a.
 - b.
 - c.
4. Specification and possible representations/implementations of "non-negative integers" ADT.
 - a. The four base (defining) operations (procedures) :
 - i. `zero` `iszero?` `succ` `pred` (may be undefined for input zero)
 - b. Sample derived operation: `add`. Write it in terms of the base operations:
 - c. Implementation 1: Unary. $\lceil 0 \rceil = '()$ $\lceil n+1 \rceil = (\text{cons } \#t \lceil n \rceil)$. How to define operations in Scheme? Other implementations are in the online slides. Due to time constraints, will not do them in class.

5. Aggregate data types:
 - a. Arrays
 - b. Records
 - c. Union types
6. **define-datatype**. A way to define new (possibly recursive) "record" types with type-checking for the fields.
 - a. **define-datatype** creates constructors for immutable variant records.
 - b. Constructors check the types of the fields and report an error if incorrect
 - c. **cases** is used to get references to the various fields.
 - d. Details of syntax for defining and using datatypes are in the slides.
 - e. We examine datatypes for binary trees, s-lists, lambda-calculus expressions.
7. Code is data. In Scheme, both have the same form. `eval` treats code as data. Don't use it in your interpreter project code!
8. A datatype for lambda-calculus expressions (you will extend this definition to include other Scheme syntax).
9. How will the `app-exp` variant of the `expression` datatype change if we allow any number of arguments in a procedure application?

```
(define-datatype expression expression?
  [var-exp
    (id symbol?)]
  [lambda-exp
    (id symbol?)
    (body expression?)]
  [app-exp
    (rator expression?)
    (rand expression?)])
```