

Class 2

Summary

An ADT hides the implementation. A list ADT can contain many different elements and has different methods to operate on those methods.

Abstract data Types

List ADT's:

- $L = \{l_1, l_2, l_3, l_4\}$ is a list of length 4
- Anything can be on the list
- In a particular setting, may be restricted (strongly typed language)
- Basic operations: $\text{First}(L)$ -> the first element of L $\text{rest}(L)$ -> a list just like L with $\text{first}(L)$ removed $\text{add}(x, L)$ -> a list with x followed by contents of L

d

eg.) input a list L of numbers

```
sum <- 0
while 1 is not empty {
  sum <- sum + first(L)
  L <- rest(L)
}
// here: sum is the sum of elements that were in L
```

Need operation: $\text{empty}(L)$ -> true if L is empty, false otherwise

eg.) input a list L

```
L2 <- a new empty list
while L is not empty {
  L2 - add(first(L), L2)
  L <- rest(L)
}
// here: L2 is the reverse of L's initial contents
//      add() adds to the front
```

Other list operations:

- Get or set the i th element
- Insert x at location i
- Delete the i th element
- Find x in L
- $\text{append}(L_1, L_2)$ -> list with contents of L_1 followed by contents of L_2
- $\text{map}(F, L)$ -> list consisting of $f(x)$ for each x in L eg $\text{map}(\text{square}, [1,2,3]) = [1,4,9]$