Collection Types

Arrays and Lists (Sequences)

2-14-2011
Opening Discussion

- Turning in assignments.
- Minute essay comments
  - Can recursion be used in non-mathematical applications?
  - What would happen if we used ! for factorial?
  - What other types are there?
  - How do our recursive functions know to keep going instead of just doing one call?
  - Why not to use BigInt for everything.
Need for Collections

- Computers are good at dealing a lot of data. So far we can only store one value in each variable. This is a significant limitation.
- Collections are types that can store multiple data values.
- Allow us to remember many things to work on.
- The collection libraries in a language are very significant.
- Scala has great collections.
The two most basic collection types in Scala are arrays and lists.

We can make either by following the type name with a parenthesized list of elements.

Can create an “empty” array using new.

Can build Lists with :: operator. Nil is empty.

Comparison

- Arrays are mutable, but fixed in size.
- Lists are immutable, but it is easy to add an element and get a new list.
You should notice that when we make an array or a list, the type is followed by square brackets.

These types are parametric. So they take type arguments.

In Scala, type parameters are placed in square brackets.
Using Arrays

- We can get to the elements in an array by putting an index in parentheses. The index is 0-referenced.
  - arr(5)
- This syntax can be used in expressions to read values.
- It can also be used in assignments to store values in the array. This is what it means to be mutable.
- Let's look at some examples of this.
Using Lists

- You can do direct access on lists, but it is inefficient.
- The better method is to use the head and tail methods.
- The elements in a list can't be changed. However, you can efficiently add new elements at the front of the list.
What are some examples of uses of arrays or lists?

Remember to turn in assignment #1 by midnight.

The third quiz is on Wednesday.