Match and Patterns

2-15-2012

Opening Discussion

- Minute essay comments:
  - Can't do “real” problems.
  - Books I read and music.
  - Functions for imaginary numbers.
  - We will learn graphics.
  - Wiggle room on input.
  - Loops would be nice.
  - Using other concepts (like loops) on IcPs.
  - Turning a String to an Int.
  - Distinction between println and return.

More

- Entering multiple types of data on the same line.
- More time coding than it would take to do by hand.
- Pulling apart tuples.
- Getting vi coloring in other places.

- Submitting the assignment?

Tracing Recursion

- The act of running through code line by line to see what it does is called tracing. It is a very important skill for programmers.
- Tracing often involves writing out variables and tracing how they change or drawing “pictures”.
- I want to show you an approach to tracing recursive functions.

Deep Recursion

- The problem in dealing with 10,000 numbers.
- Rewrite the code so it doesn't have to remember things.

The match Construct

- Scala has a second conditional called match.
  - *expr* match {
    - case *pattern1* => *expr*
    - case *pattern2* => *expr*
    - ...
  - }
- The first case that matches is evaluated.
- Can put if-guards.

Patterns

- Matches a value to a form.
- Form can include tuples and many other things.
- Literals and names starting with capital letters have to match values.
- Names starting with lower case letters are bound as new values.

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.

Sequences

- One variable/name, many values.
- Integer indexes starting with 0.
- Our first examples are Lists and Arrays.

Basic Arrays and Lists

- 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.

Parametric Types

- 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.
- Lists work very well with recursion.

List and Array Patterns

- You can make patterns with Lists and Arrays.
- For Arrays:
  - Array(1,2,a,b,c)
- For Lists:
  - List(1,2,a,b,c)
  - h::t - matches any non-empty list
  - Nil - matches an empty list

Minute Essay

- Questions?
- Quiz #3 is on Friday.