

Loops

2-23-2011

Opening Discussion

- Minute essay comments
 - Length of `lcp` solutions.
 - Getting input into `lcp`s.
 - What does `tabulate` do?
 - Will `foldLeft` be on the exam?

while Loop

- Recursion is sufficient for making repetition, but in imperative languages it isn't the normal approach. Instead, people use loops.
- The simplest loop is the while loop.
 - *while(condition) statement*
- The condition is evaluated first. If it is true the statement (possibly a block) executes.
- This repeats until the condition is false.

do-while Loop

- The partner to the while loop is the do-while loop.
 - do {
 - *statement*
 - } while(*condition*)
- This loop is post-check instead of the pre-check of the normal while loop.
- Always happens once.
- The while loop might never happen.

The for Loop

- The most commonly used loop in most languages is the for loop. The Scala version is a bit different from most.
- Often used for counting:
 - `for(i <- 1 to 10) { ... }`
- In general it is a “for each” loop that goes through a collection.
 - `for(e <- coll) { ... }`
- Variable takes on value of each element in the collection.

Range Type

- Range types provide an easy way to make collections for counting.
- “to” and “until” operate on numeric types to produce ranges.
 - 1 to 10
 - 0 until 10
- Use “by” to change the stepping in a range.
 - 1 to 100 by 2
 - 10 to 1 by -1
 - 'a' to 'z' by 3

yield

- The for loop can be used as an expression if you put `yield` between the end of the for and the expression after it.
 - `for(e <- coll) yield expr`
- What you get back will be a collection that is generally of the same type as what you iterated over.

if Guards

- You can put conditions in the for that will cause some values to be skipped.
 - `for(n <- nums; if(n%2==0)) ...`

Multiple Generators

- You can also put multiple generators in a for loop.
 - `for(i <- 1 to 10; j <- i to 10) ...`
- You can combine as many generators and guards as you want. You can also declare variables in the middle of the for.
- The thing you assign into is like a `val` so it can be a “pattern”. We have only seen this with tuples so far.

Multidimensional Arrays

- You can have collections of collections. A common example would be something like `Array[Array[Double]]` to represent a matrix.
- Both `fill` and `tabulate` can be used to make these.
 - `val ident=Array.tabulate(3,3)((i,j) => if(i==j) 1.0 else 0.0)`

Views

- This is an advanced topic, but can be significant for performance.
- When you map or filter a normal collection, it runs through the whole thing and makes a new collection. Doing a lot of these in a row can be inefficient.
- A view is a non-strict form of a collection. Doing map or filter doesn't produce a new one. It only does the work when really needed.

Minute Essay

- Any questions?
- Midterm is on Friday. The review session will be 5:00-6:00pm. If you can't make this you can always send e-mail or see me some other time.