Reading for next time: LCR 4.

Program Development:

Notice we can build programs bottom-up: Start with what we are given (e.g., forward, backward, motors, stop), and we use to construct more useful functions, such as yoyo and wiggle. And then we use these to construct even more complex behaviors such as the dance routine.

You don't put things together randomly; you have to have an idea what higher level behaviors will be useful.

You can also work top-down, which means that you start with what you want (your goal) and you figure out what you need to implement that. And then you figure what you need to implement those things. And keep working downward, until you get to things you have already got (i.e., already implemented).

Repetition in C++

There are two primary mechanisms for repetition in C++ (and most other programming languages, although the syntax may be different): definite iteration and indefinite iteration.

With definite iteration you know in advance how many times you are going to do something. The primary way to do this in C++ is a *for-loop*.

```
for ( <initialization> ; <continuation> ; <incrementation> ) {
    <statements>
}
```

For example, to call dance() ten times:

```
for ( int step = 1; step <= 10; step = step + 1 ) {
   dance();
}
```

This is what a for-loop does:

(1) do the <initialization> (usually declare and initialize a variable).
(2) test the <continuation> condition. If it's satisfied, continue with step (3), otherwise the loop is done (step 6).
(3) do the loop body (the <statements>).
(4) do the <incrementation> (usually adds one to the loop variable).
(5) go back to step (2).
(6) we're done with the loop; continue with the rest of the program.

A general rule of program development: Test the boundary conditions (e.g., the smallest, largest, or other "special" values).

Indefinite Iteration

When we don't know the exact number of times, but we are going to iterate so long as some condition is true.

```
while ( <condition> ) {
   <statements>
}
```

As long as the <condition> is true, it will keep repeating the <statements>. This is called a leading-decision loop, because the test is done at the beginning of the loop, so if it's initially unsatisfied, the loop isn't repeated at all.