Limit Bounds

A limit loop first does a calculation, and then checks to see if the calculation has reached a limit. Limit loops are used extensively in scientific, financial and graphical computing. Kinds of limits include reduction to zero, bisection, non-convergence tests (as in Calculus) and successive approximations. You'll use several of those in Lecture.

Here is a limit loop whose limit is the reduction to zero. This computes the sum of the digits in an integer, `n`:

```
int temp = n;
while (temp > 0)
{
    sum += temp % 10;
    temp /= 10;
}
cout << "The sum of the digits in " << n << " is " << sum << endl;
```

- The expression `temp % 10` always returns the last digit in the positive integer `temp`.
- The expression `temp / 10` returns the number without its final digit.

In this case, the condition that is being tested is the value of `temp`, which is changed each time through the loop. Once `n` reach its **limit** (`0`), the loop terminates.

> *Note that instead of changing `n` itself, the solution creates a separate variable named `temp`. This is better style because it doesn't confuse the concept of a parameter or input variable, with that of a local variable used for the function output. In this case, if the loop had used `n` then the last output statement would have been incorrect.*

More Loop Plans

Here are the plans that I created for each of the remaining two functions. Using limit loops, you should be able to complete these on your own.

Here is `firstDigit()`

```
// result <- |n|
// while result greater or equal to 10
//      result <- result / 10
// return result;
```

And, here is `numberOfDigits`

```
// Assume that 0 is a digit
// result <- |n|
// counter <- 1
// while result is greater or equal to 10
//     result <- result / 10
//     counter <- counter + 1
// return counter
```