COMP161
Counted Loops
Spring 2017

In this worksheet you'll practice reading, writing, and analyzing loops that count.

Some Vocabulary

A COUNTED LOOP is a loop that works to count out some set of numbers. Alternatively, we can say that the loop does a TRAVERSAL of this set. As we've seen, these loops typically work by modifying the state of a counter variable such that over the course of the loop that variable takes on the value of every number from the desired set of numbers.

The code within the curly brackets of the loop is called the LOOP BODY. A single execution of the loop body is called an ITERATION of the loop. The amount that the counter is updated by from one iteration to the next is called the STRIDE[1].

[1] Often we'll talk in terms of the stride as the absolute difference as counting up (positive stride) or down (negative stride) is typically an option

Practice by Doing

A good way to practice code is to actually code. Here we're simply concerned with loops so the trick is to find a simple task to loop that lets us verify the correctness of our loop. The traditional option is write a loop that prints your counter values. In Figure 1 we see a loop that we can drop inside a *main* procedure such that the resultant program will give us visual confirmation that our loop counts through the interval $[0, 7)$. Choosing a smallish value for n makes verifying the output easy while ensuring the loop is definitely going through several iterations. We could then go change the value of n to see how the loop behaves as n changes[2]

[2] What happens if n is zero? What if n is negative?

```
int n7; int io; while( i < n ) std::cout « i « ' '  std::cout « '';
```

Figure 1: Testing that the loop counts to n by output

There are ways to write gTest tests to verify our loop, but we'll save those for when we have a few more tricks up our sleeve.

Write all the loops requested by worksheet in a single main procedure as shown above. Use different variables for each loop.

1. Counting the interval $[0, n)$

   (a) Write a *for* loop that counts out the interval $[0, n)$.

(b) Write a *do-while* loop that counts out the interval $[0, n)$. Will this loop work exactly the same as your for loop for every value of n?

(c) Write a *while* loop that counts backwards through the interval.

2. Other Counting

(a) Write a *for* loop that counts down through the first n multiples of 5.

(b) Write a *while* loop that counts up through the first n multiples of 7 but uses a stride of 1. (Hint: Notice that we're counting iterations now. So on the i^{th} iteration use i to compute the i^{th} multiple.)

(c) Write two *do-while* loop that counts up through the even numbers in the interval $[2n, 5n)$. The first should count by setting the counter directly, and the second should count number of iterations while computing the desired numbers[3].

[3] the second has a stride of 1 and the first does not

(d) Write a loop of your choosing that counts up through the first n powers of 2 such that the value of your counter is the i^{th} power on the i^{th} iteration.

(e) Write a loop of your choosing that counts down through the first n powers of 3 such that the value of your counter is the i^{th} power on the i^{th} iteration.

3. Some of these things are not like the others.

(a) What is the stride for all the loops your wrote except the last two?

(b) How would you characterize the stride on the final two loops?

(c) If I started at 1 and counted to n using the powers of two stride pattern, how many iterations would I need to reach n? (Hint: Start by picking some n that are powers of two and count it out. See if you can find a pattern.)