The `n_chars()` function takes two arguments: a character `c` and an integer `n`. It then uses a loop to display the character the number of times the integer specifies:

```cpp
while (n-- > 0) // continue until n reaches 0
  cout << c;
```

Notice that the program keeps count by decrementing the `n` variable, where `n` is the formal parameter from the argument list. This variable is assigned the value of the `times` variable in `main()`. The `while` loop then decreases `n` to zero, but, as the sample run demonstrates, changing the value of `n` has no effect on `times`.

**Another Two-Argument Function**

Let's create a more ambitious function, one that performs a nontrivial calculation. Also, the function will illustrate the use of local variables other than the function's formal arguments.

Many states in the United States now sponsor a lottery with some form of Lotto game. Lotto lets you pick a certain number of choices from a card. For example, you might get to pick six numbers from a card having 51 numbers. Then, the Lotto managers pick six numbers at random. If your choice exactly matches theirs, you win a few million dollars or so. Our function will calculate the probability that you have a winning pick. (Yes, a function that successfully predicts the winning picks themselves would be more useful, but C++, although powerful, has yet to implement psychic faculties.)

First, we need a formula. Suppose you have to pick six values out of 51. Then, mathematics says you have one chance in \( R \) of winning, where the following formula gives \( R \):

\[
R = \frac{51 \times 50 \times 49 \times 48 \times 47 \times 46}{6 \times 5 \times 4 \times 3 \times 2 \times 1}
\]

For six choices the denominator is the product of the first six integers, or six factorial. The numerator also is the product of six consecutive numbers, this time starting with 51 and going down. More generally, if you pick `picks` values out of `numbers` numbers, the denominator is `picks` factorial and the numerator is the product of `picks` integers starting with the value `numbers` and working down. You can use a `for` loop to make that