`print_count` has one parameter, `n`, of type `int`. It returns nothing, so I've specified `void` as the return type and omitted the `return` statement. Since `print_count` doesn't return a value, we can't call it in the same way we call `average`. Instead, a call of `print_count` must appear in a statement by itself:

```
print_count(i);
```

Here's a program that calls `print_count` 10 times inside a loop:

countdown.c

```
/* Prints a countdown */

#include <stdio.h>

void print_count(int n)
{
  printf("T minus %d and counting\n", n);
}

int main(void)
{
  int i;

  for (i = 10; i > 0; --i)
    print_count(i);

  return 0;
}
```

Initially, `i` has the value 10. When `print_count` is called for the first time, `i` is copied into `n`, so that `n` takes on the value 10 as well. As a result, the first call of `print_count` will print

```
T minus 10 and counting
```

`print_count` then returns to the point at which it was called, which happens to be the body of a `for` statement. The `for` statement resumes where it left off, decrementing `i` to 9 and testing whether it's greater than 0. It is, so `print_count` is called again, this time printing

```
T minus 9 and counting
```

Each time `print_count` is called, `i` is different, so `print_count` will print 10 different messages.

PROGRAM

Printing a Pun (Revisited)

Some functions have no parameters at all. Consider `print_pun`, which prints a bad pun each time it's called:

```
void print_pun(void)
{
  printf("To C, or not to C: that is the question.\n");
}
```