Step right up to the counter...and accumulate!

In programming, especially in loops, we take advantage of the loop index for counting (this is too obvious because the loop index counts for us!) and accumulating(adding up numeric data, or subtracting, multiplying etc.).

This is illustrated in several programs in the textbook in chapters 4 and 5.
The design for the addition (or subtraction) accumulator is as follows:

1. Set up a variable to accumulate the data, and set it *initially* to the value zero.

2. Within the loop add or subtract from the accumulator.

A facsimile (with my style of naming variables) of program 5.2:

```
int grade_total = 0;
....

 for ( i = 1; i <= number_of_grades; ++i)
 {
      printf ("Enter Grade #%i: ",i);
      scanf ("%i", &grade);

      grade_total = grade_total + grade;

 }
```

The variable **i** is the loop index.

Notice ***grade_total*** is on **both** sides of the equal sign. This takes the *current* value of ***grade_total*** (on the right), adds *grade* to it, then stores the result in ***grade_total****.* That's it!

When the program exits the loop, the ***grade_total*** will be the total of all the grades entered during the loop process. It may seem a little fuzzy now, but once you understand the concept you will use it all the time.

Notice also the counter will be displayed as we put in the grades:

Enter Grade #**1**:   (here the value of variable i is **1**)

Enter Grade #**2**:   (here the value of variable i is **2** etc.)

At the end of the loop (i.e. outside the loop) you have the accumulated grades in ***grade_total***.

So, you could output them:

printf ("The total of grades is %i\n", grade_total);

then, to calculate an average:

ave_grade = (float) grade_total / number_of_grades;

I used the *typecast* **(float)** so that the computation would be computed as a decimal value, and if **ave_grade** is **float** type you can then print out as many decimals as you like. *Notice only one typecast is needed to make the entire calculation a float.*

For example for one decimal place:

printf ("The average of the grades is %.1f\n", ave_grade);

Simple as that!

The next topic this week deals with the concept of having a **for** loop within a **for** loop. This is known as: **insanity**. Oh, I mean: **nested for loops**. ☺