5 Loops

The simplest form of a loop, for example, to execute a block of code for subscript values from 1 to 100, is

```
for(i = 1; i <= 100; i++)
{
    body of loop
}
```

More generally,

```
for(i = start_value; i <= end_value; i = i + increment_value)
{
    body of loop
}
```
If you know exactly how many lines of data, with an input data file of perhaps

5
10.0
20.1
-310.823
99.128
40.4

you could code

scanf("%ld",&number_of_values);
for(i = 1; i <= number_of_values; i++)
{
    scanf("%lf",&this_value);
    body of code using "this_value"
}

The first line has the exact number of values, and then you read exactly that many things.
Basic Rules for for Loops

• The first expression is the initialization.
• The second expression tests whether to keep going
• The third expression says what to do at the end of each iteration

• start_value, end_value, increment_value can be positive or negative and can be any legal data type (but watch out for “exact” tests if floating point!!!!)

• If there is no iteration to be executed, then no iteration occurs

• for(i = 1; i <= 10; i++) // the i++ says ‘‘increment’’

• for(i = 1; i <= 10; i+=2) // the i += 2 says ‘‘i = i + 2’’

• It is legal to get fancy with all three expressions. This is not advisable without careful testing.

• break gets you out of the current loop

• continue says skip the rest of this loop and go on to the next
If you don’t know exactly how many lines of data, you could use as a *sentinel value* some value that you know (for some reason) can’t possibly be a valid data item. With a sentinel value, there are *several* ways to read data. Any of these is ok, if used correctly.

read1.c, read2.c, read3.c, read4.c