CSC240 - Standard Sequential File Input in C++

(1) Start by declaring the file access functions:

in the Express editions, type...

```
#include <fstream>
#include <iostream>
using namespace std;

or, in older environments,

#include <fstream.h>
```

(2) Inside the program, declare a variable to "manage" accessing the file:

```
void main( void )
{
        fstream infile;
```

(3) After declaring the variable, establish a connection to a file:

```
        infile.open( "c:\\temp\\data.txt", ios:in );
```

In general, the "open" statement looks like this:

```
        <fstream variable name>.open( <file-name>,
                                      <access-mode> );
```

For the "access mode", use the predefined constant, 'ios::in', to read from a file. There are other access modes, like 'ios::out' to write to a file.

You can specify the file name as a constant inside double-quotes, or you can use a string variable:

```
        char fileName[81];
```

Before opening the file, you must assign the variable an actual file name, by asking the user or using some other method.

If you specify the file name as a constant, you have to use the double-backslash for each back-slash in the file path description because the back-slash inside double-quotes is the special escape character. For example, recall that "\t" is a tab. So "\\" is necessary to indicate a single back-slash. Notice that this is necessary because the characters are inside double-quotes. If you ask the user to enter the file name, they do not need the double-backslash.

(4) Read the data from the file:

```
infile >> distance;
```

C++ treats data from a sequential file like a stream or a pipeline of information. The data is read in sequence, one at a time, with no skipping. Each item must be read before you can read the next one.

Reading one item from the file is just like reading one from the keyboard, but with a different source, so the rules are the same. For example, the type of the information, number or characters, must match the type of the variable. To read more than one piece of information, you need more than one file input statement. To read several lines or "records" from a file, each with the same information, you need to use a loop to repeat the process. By using one or more arrays, you can save all the information from the file for later processing.

(5) Close the connection with the file:

```
infile.close();
```

The connection to a file is automatically "closed" when the program ends, but this statement will close the connection immediately. Each connection uses system resources, so it is a good idea to close a file when the program is done reading the data. Note that closing and re-opening a connection to the same file "rewinds" it - the pipeline starts over with the first piece of information in the file.