Differences between a text file and a data file

A text file is a file that is processed character by character. In a text file, both a space and the end-of-line (also known as the newline or '\n') character are processed, since they are characters. We often add the suffix `.txt` to a text file to indicate that it is a text file, although it is not compulsory.

A data file, on the other hand, is a file from which we want to read (extract) data values, such as e.g. integers, double or float values, strings or characters. Data files usually have the suffix `.dat`.

This means that we will process them in different ways. When processing a text file, we want to process every character in the file, including the spaces and end-of-line characters. We use the `get` member function for streams to read one character at a time from a text file, and use the `eof` member function to explicitly test whether all of the file has been read. The `eof` member function returns `true` when the program has read past the end of the file, which is why we use `! eof` in the `while` loop to process a text file.

A text file is typically processed in the following way (Assume that the internal file name for the text file is `infile`.):

```
char ch;
infile.get(ch);       //read one character
while (! infile.eof())//make sure we have not read past
                       // the end of the file – note the '!'
{
      //process ch       //process the character that has been read
      infile.get(ch);//read another character to process
}
```

When processing a data file, we are processing data values, and the process of reading the values (extracting) from the data file, works much the same way as reading data from the keyboard. Therefore we (or rather the program we write) will ignore spaces and end-of-line characters when reading data values from the file, similar to how data is input or read from the keyboard.

The typical way to read a file containing values that should be processed as `int`, `string` or `float` e.g. a file containing one `int` followed by a `string` on each line, follows:

```
int value;
string name;
while (infile >> value >> name)
{
//process value and name
}
```

The extraction operator >> will read values for variables `value` and `name` from the data file infile, *and* at the same time test whether the file still contains values to be read. While there are still values to be read, the loop will be repeated. As soon as the last value is read and the end of the file is reached, the condition `(infile >> value >> name)` will become false and the loop will be exited.

Note: Students are sometimes under the impression that a data file should be read character by character or that each line in the data file should be read as a string, which is then converted to the data values required. This is unnecessary, as can be seen from the example above.