Return to Topic Menu | Computer Science Main Page | MathBits.com | Terms of Use | Resource CD

End of File Function

So, just how much data is in that file? The exact contents of a file may not be precisely known. Usually the general format style of the file and the type of data contained within the file are known.

The amount of data stored in the file, however, is often unknown. So, do we spend our time counting data in a text file by hand, or do we let the computer deal with the amount of data? Of course, we let the computer do the counting.

C++ provides a special function, **eof( )**, that returns nonzero (meaning TRUE) when there are no more data to be read from an input file stream, and zero (meaning FALSE) otherwise.

Rules for using end-of-file (eof( )):

1. Always test for the end-of-file condition **before** processing data read from an input file stream.
   a. use a priming input statement before starting the loop
   b. repeat the input statement at the bottom of the loop body
2. Use a while loop for getting data from an input file stream. A for loop is desirable only when you know the exact number of data items in the file, which we do not know.

```
#include <iostream.h>
#include <fstream.h>
#include <assert.h>

int main(void)
{
    int data;          // file contains an undermined number of integer values
    ifstream fin;      // declare stream variable name

    fin.open("myfile",ios::in);    // open file
    assert (!fin.fail( ));
```

```
    fin >> data;        // get first number from the file (priming the input statement)
                        // You must attempt to read info prior to an eof( ) test.
    while (!fin.eof( ))      //if not at end of file, continue reading numbers
    {
        cout<<data<<endl;    //print numbers to screen
        fin >> data;         //get next number from file
    }
    fin.close( );       //close file
    assert(!fin.fail( ));
    return 0;
}
```

The eof( ) function has been known to be persnickety under certain conditions. If you experience problems with the function, you may want to consider this alternate approach to check for end of file:

```
//This example creates a file of apstrings
//then opens the new file and prints the info to the screen
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>              //for exit()
#include "apstring.cpp"

int main(void)
{
    ofstream fout;
    ifstream fin;
    apstring sentences, sent;

    fout.open("sentences.dat");  //creating the file
    if (!fout)
    {
        cerr<<"Unable to open file"<<endl;
        exit(1);
    }
    for(int i = 0; i < 5; i++)              //file will contain 5 apstring variables
    fout<<"This is sentence #"<<i+1<<endl;
    fout.close( );
   //open file and read from file
    fin.open("sentences.dat");      //open file to access information
    while (getline(fin,sent))   //The test condition is TRUE
                                  // only while there is something to read.
    {                            //Works nicely as an end of file check.
        cout<<sent<<endl;
    }
    fin.close( );
    return 0;
}
```