

# Chapter 9

## File Streams

3rd Edition

Computing Fundamentals with C++

Rick Mercer

Franklin, Beedle & Associates

# Goals

- Use `ifstream` objects for disk file input
- Use `ofstream` objects for disk file output
- Apply the indeterminate loop pattern to process data until end of file

# ifstream objects

- `ifstream` objects
  - allow input from a disk file
  - are similar to the `istream` object named `cin`
    - both share the same named operations
    - both use the same operator for input `<<`
  - must be initialized by the programmer
  - can be tested to determine if a disk file actually exists

# ifstream objects

- General form to construct ifstream objects

```
ifstream object-name ( "file-name" ) ;
```

- This associates the object name `inFile` with the file named `"myfile.txt"` in the working folder

```
ifstream inFile("myfile.txt");
```

# Read 3 different types

```
#include <fstream> // for class ifstream
#include <iostream>
using namespace std;
int main() {
    int n;
    double x;
    string str;
    ifstream inFile("input.txt");
    cout << "Good? " << inFile.good() << endl;

    // Read 3 different types from "input.txt"
    inFile >> n;
    inFile >> x;
    inFile >> str;

    cout << "  n: " << n << endl;
    cout << "  x: " << x << endl;
    cout << "str: " << str << endl;
    return 0;
}
```

*input.txt*

```
100
99.9
Dakota
```

*Output*

```
Good? 1
  n: 100
  x: 99.9
str: Dakota
```

# Getting the Path Right

- It is easy to initialize an `ifstream` object and not have it associated with an actual disk file
  - perhaps the file does not exist
  - perhaps the path is wrong
  - perhaps you used `\` to separate folder names
- Recall escape sequences look like like `\n` `\t`
- `\` is also used to separate paths *DOS and Windows (Unix uses / so this is not an issue in the Unix environment):*

```
ifstream inFile("c:\myc++\input.dat");    // NO!  
ifstream inFile("c:\\myc++\\input.dat");  // YES
```

# Reading File names

- However, when the user enters the file name, they may use one \, or in Unix, one /

```
string filename;  
cout << "Enter file name: ";  
cin >> filename;  
ifstream inFile(filename);
```

- Dialogue in DOS/Windows

```
Enter file name: c:\temp\in.dat
```

- Dialogue in Unix

```
Enter file name: /temp/in.dat
```

# Indeterminate Loop with a File

- The *end of file* event can be used to terminate user input

```
cout << "Enter numbers or end of file to quit\n";
cout << "ctrl-D (UNIX)" << endl;
cout << "ctrl-Z (DOS or Windows) " << endl;
string str;
int n = 0;

while (cin >> str)
    n++;

cout << "You made "
    << n << " entries";
```

```
Enter numbers or end of file to quit
ctrl-D (UNIX)
ctrl-Z (DOS or Windows)
a
b
You made 2 entries
```



# Processing until End of File

- We often need to extract data from files stored on a computer disk
- We use `ifstream` objects for this kind of input
- The next slide shows a program that averages numbers from the file `c:\input.dat`

# Read until end of file

```
ifstream inFile("input.data");  
double number, sum = 0.0;  
int n = 0;  
while (inFile >> number) {  
    sum += number;  
    n++;  
}  
cout << "Average: " << (sum / n) << endl;
```

*input.data*

70.0
80.0
90.0
75.0
85.0

*Output*

Average: 70
-------------

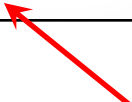
# Mixing Numbers and Strings

- Use care when input has a mix of numeric and alphanumeric data. In this code, the first `cin` fails

```
double hours, rate;
int exemptions;
string maritalStatus;
string lastName, firstName;
ifstream inFile("payroll.txt");
while (inFile >> hours >> rate
       >> maritalStatus >> exemptions
       >> firstName >> lastName) {
```

```
40.0 8.88 3 S Taylor Cook
42.0 7.77 2 M Kelsey Woodman
```

Unexpected string to int  
destroys the input stream



- This is an attempt to store a string into an int
  - Need to swap `maritalStatus` with `exemptions`

# Indeterminate Loop with More Complex Disk File Input

- Problem: Count the words in a book
- Algorithm
  - For each line in the input file:
    - Determine the number of words in that line and add to the running sum
- We'll need a loop inside that loop to count the number of words in that line
- Each line will be processed until end of file in a loop
  - We need the `getline` function (next slide)

# The getline Function

- Use `getline` to read in lines of data as one string (with no 3<sup>rd</sup> argument, the default eol is `'\n'`)

```
istream & getline(istream & is, string & s,  
                 char sentinel = '\n')
```

- Examples

```
string name, address;  
getline(cin, address); // Read from keyboard  
getline(inFile, name, '.'); // Read from file.
```

```
// Read until end of line  
while( getline(infile, address) ) {  
    // . . .  
}
```

# Nested loops

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

int main() {
    string line;
    ifstream inFile("taleOf2.txt");
    int words = 0;

    while (getline(inFile, line)) {
        // Counting spaces needs one more word per line
        words++;
        for(int index = 0; index < line.length(); index ++) {
            if(line[index] == ' ')
                words++; // assume 1 space separates all words
        }
    }
    cout << words; // 120
    return 0;
}
```

*taleOf2.txt*

It was the best of times,  
it was the worst of times,  
it was the age of wisdom,  
it was the age of foolishness,  
it was the epoch of belief,  
it was the epoch of incredulity,  
it was the season of Light,  
it was the season of Darkness,  
it was the spring of hope,  
it was the winter of despair,  
we had everything before us,  
we had nothing before us,  
we were all going direct to Heaven,  
we were all going direct the other way--  
in short, the period was so far like the present period,  
that some of its noisiest authorities insisted on its being received,  
for good or for evil, in the superlative degree of comparison only

# ofstream objects

- The files storing large amounts of data are typically created by programs that send output to those files *rather than the screen*
- class `ofstream` represents a disk file for output
- General form:  

```
ofstream object-name ( "file-name" );
```

# ofstream

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
    ofstream outFile("output.txt");
    outFile << "This does not go to the screen" << endl;
    cout << "This does" << endl;
    double x = 1.23;
    int n = -1;
    string str = "A string";

    outFile << x << endl;
    outFile << n << endl;
    outFile << str << endl;
    outFile.close();
    return 0;
}
```

*output.txt*

```
This does not go to the screen
1.23
-1
A string
```