Chapter 12: Advanced File Operations

Kafi Rahman
Assistant Professor
Truman State University

Using Files

- Requires fstream header file
  - use ifstream data type for input files
  - use ofstream data type for output files
  - use fstream data type for both input, output files
- Can use >>, << to read from, write to a file
- Can use eof member function to test for end of input file

Default File Open Modes

- ifstream:
  - open for input only
  - file cannot be written to
  - open fails if file does not exist
- ofstream:
  - open for output only
  - file cannot be read from
  - file created if no file exists
  - file contents erased if file exists
- or, we can use fstream object to do both

More File Open Details

- Can use filename, flags in definition:
  - ifstream infile("grades.txt");
- File stream object set to 0 (false) if open failed:
  - if (!infile) …
- Can also check fail member function to detect file open error:
  - if (infile.fail() == true) …
  - if(infile.is_open() == false) …

fstream Object

- fstream object can be used for either input or output
- Must specify mode on the open statement
- Sample modes:
  - ios::in – input
  - ios::out – output
- Can be combined on open call:
  - dFile.open("class.txt", ios::in | ios::out);

Using Files - Example

```
// copy 10 numbers between files
// open the files
fstream infile("input.txt", ios::in);
fstream outfile("output.txt", ios::out);

for (int i = 1; i <= 10; i++)
{   int num;
    infile >> num;    // use the files
    outfile << num;
}
infile.close();      // close the files
outfile.close();
```

12.2

File Output Formatting

File Output Formatting

- Use the same techniques with file stream objects as with cout: showpoint, setw(x), showprecision(x), etc.

- Requires iomanip to use manipulators

File Output Formatting (cont)

Program 12-3

```
1  // This program uses the setprecision and fixed
2  // manipulators to format file output.
3  #include <iostream>
4  #include <iomanip>
5  #include <fstream>
6  using namespace std;
7
8  int main()
9  {
10    fstream dataFile;
11    double num = 17.816392;
12
13    dataFile.open("numfile.txt", ios::out);   // Open in output mode
14
15    dataFile << fixed;             // Format for fixed-point notation
16    dataFile << num << endl;       // Write the number
17
18    dataFile << setprecision(4);   // Format for 4 decimal places
19    dataFile << num << endl;       // Write the number
20
21    dataFile << setprecision(3);   // Format for 3 decimal places
22    dataFile << num << endl;       // Write the number
23
```

File Output Formatting (cont)

```
24      dataFile << setprecision(2);  // Format for 2 decimal places
25      dataFile << num << endl;      // Write the number
26
27      dataFile << setprecision(1);  // Format for 1 decimal place
28      dataFile << num << endl;      // Write the number
29
30      cout << "Done.\n";
31      dataFile.close();             // Close the file
32      return 0;
33  }
```

Contents of File `numfile.txt`

```
17.816392
17.8164
17.816
17.82
17.8
```

12.3

Passing File Stream Objects to Functions

Passing File Stream Objects to Functions

- It is very useful to pass file stream objects to functions
- Be sure to always pass file stream objects by reference

Passing File Stream Objects to Functions (cont)

Program 12-5

```
1  // This program demonstrates how file stream objects may
2  // be passed by reference to functions.
3  #include <iostream>
4  #include <fstream>
5  #include <string>
6  using namespace std;
7
8  // Function prototypes
9  bool openFileIn(fstream &, string);
10 void showContents(fstream &);
11
12 int main()
13 {
14      fstream dataFile;
15
16      if (openFileIn(dataFile, "demofile.txt"))
17      {
18           cout << "File opened successfully.\n";
19           cout << "Now reading data from the file.\n\n";
20           showContents(dataFile);
21           dataFile.close();
22           cout << "\nDone.\n";
23      }
```

Passing File Stream Objects to Functions (cont)

```
24    else
25        cout << "File open error!" << endl;
26
27    return 0;
28 }
29
30 //*********************************************************
31 // Definition of function openFileIn. Accepts a reference   *
32 // to an fstream object as an argument. The file is opened  *
33 // for input. The function returns true upon success, false *
34 // upon failure.                                            *
35 //*********************************************************
36
37 bool openFileIn(fstream &file, string name)
38 {
39    file.open(name, ios::in);
40    if (file.fail())
41       return false;
42    else
43       return true;
44 }
45
46 //*********************************************************
47 // Definition of function showContents. Accepts an fstream  *
48 // reference as its argument. Uses a loop to read each name *
49 // from the file and displays it on the screen.             *
50 //*********************************************************
```

Passing File Stream Objects to Functions (cont)

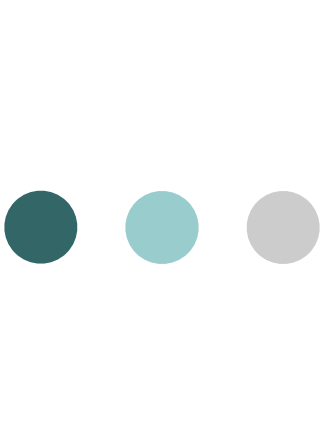
```
51
52 void showContents(fstream &file)
53 {
54    string line;
55
56    while (file >> line)
57    {
58       cout << line << endl;
59    }
60 }
```

Program Output

```
File opened successfully.
Now reading data from the file.

Jones
Smith
Willis
Davis

Done.
```



12.4

More Detailed Error Testing

More Detailed Error Testing

- Can examine error state bits to determine stream status
- Bits tested/cleared by stream member functions

ios::eofbit	set when end of file detected
ios::failbit	set when operation failed
ios::badbit	set when invalid operation attempted
ios::goodbit	set when no other bits are set

Member Functions / Flags

eof()	true if eofbit set, false otherwise
fail()	true if failbit or hardfail set, false otherwise
bad()	true if badbit set, false otherwise
good()	true if goodbit set, false otherwise
clear()	clear all flags (no arguments), or clear a specific flag

Member Functions / Flags

```
68  void showState(fstream &file)
69  {
70     cout << "File Status:\n";
71     cout << "  eof bit: " << file.eof() << endl;
72     cout << "  fail bit: " << file.fail() << endl;
73     cout << "  bad bit: " << file.bad() << endl;
74     cout << "  good bit: " << file.good() << endl;
75     file.clear();  // Clear any bad bits
76  }
```

12.5

Member Functions for Reading and Writing Files

Member Functions for Reading and Writing Files

- Functions that may be used for input with whitespace, to perform single character I/O, or to return to the beginning of an input file
- Member functions:
  - getline: reads input including whitespace
  - get: reads a single character
  - put: writes a single character

The getline Function

- Three arguments:
  - Name of a file stream object
  - Name of a string object
  - Delimiter character of your choice
  - Examples, using the file stream object myFile, and the string objects name and address:
  - `getline(myFile, name);`
  - `getline(myFile, address, '\t');`

  - If left out, '\n' is default for third argument

The getline Function

Program 12-8

```
1  // This program uses the getline function to read a line of
2  // data from the file.
3  #include <iostream>
4  #include <fstream>
5  #include <string>
6  using namespace std;
7
8  int main()
9  {
10    string input;     // To hold file input
11    fstream nameFile; // File stream object
12
13    // Open the file in input mode.
14    nameFile.open("murphy.txt", ios::in);
15
16    // If the file was successfully opened, continue.
17    if (nameFile)
18    {
19        // Read an item from the file.
20        getline(nameFile, input);
21
```

The getline Function

```
22          // While the last read operation
23          // was successful, continue.
24          while (nameFile)
25          {
26             // Display the last item read.
27             cout << input << endl;
28
29             // Read the next item.
30             getline(nameFile, input);
31          }
32
33          // Close the file.
34          nameFile.close();
35       }
36       else
37       {
38          cout << "ERROR: Cannot open file.\n";
39       }
40       return 0;
41 }
```

Program Output

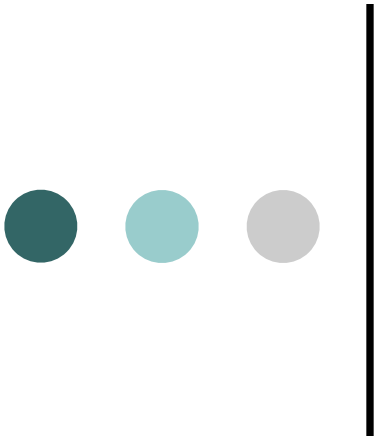
```
Jayne Murphy
47 Jones Circle
Almond, NC 28702
```

Single Character I/O

- get: read a single character from a file
  - char letterGrade;
  - infile.get(letterGrade);
  - Will read any character, including whitespace
- put: write a single character to a file
  - outfile.put(letterGrade);

Single Character I/O

- Demo program
  - review mulfile.cpp program



12.6

Working with Multiple Files

Working with Multiple Files

- Can have more than one file open at a time in a program
- Files may be open for input or output
- Need to define file stream object for each file

Working with Multiple Files

Program 12-12

```
1  // This program demonstrates reading from one file and writing
2  // to a second file.
3  #include <iostream>
4  #include <fstream>
5  #include <string>
6  #include <cctype> // Needed for the toupper function.
7  using namespace std;
8
9  int main()
10 {
11    string fileName;     // To hold the file name
12    char ch;             // To hold a character
13    ifstream inFile;     // Input file
14
15    // Open a file for output.
16    ofstream outFile("out.txt");
17
18    // Get the input file name.
19    cout << "Enter a file name: ";
20    cin >> fileName;
21
22    // Open the file for input.
23    file.open(name, ios::in);
24
25    // If the input file opened successfully, continue.
```

Working with Multiple Files

```
26     if (inFile)
27     {
28         // Read a char from file 1.
29         inFile.get(ch);
30
31         // While the last read operation was
32         // successful, continue.
33         while (inFile)
34         {
35            // Write uppercase char to file 2.
36            outFile.put(toupper(ch));
37
38            // Read another char from file 1.
39            inFile.get(ch);
40         }
41
42         // Close the two files.
43         inFile.close();
44         outFile.close();
45         cout << "File conversion done.\n";
46     }
47     else
48        cout << "Cannot open " << fileName << endl;
49     return 0;
50 }
```

Working with Multiple Files

Program Screen Output with Example Input Shown in Bold

```
Enter a file name: hownow.txt [Enter]
File conversion done.
```

Contents of `hownow.txt`

```
how now brown cow.
How Now?
```

Resulting Contents of `out.txt`

```
HOW NOW BROWN COW.
HOW NOW?
```

Appending to a file

```
int main()
{
    fstream outfile;
    outfile.open("data.txt", ios::app);

    // if the file does not exist, it will be created

    cout <<"Appending new data to the file " << endl;
    outfile << "Ferrari, 2021, Justin Trudeau" << endl;

    cout <<"Done" << endl;

    outfile.close();
}
```