Streams and File I/O

File Operations

- File: a set of data stored on a computer, often on a disk drive
- Programs can read from, write to files
- Used in many applications:
  - Word processing
  - Databases
  - Spreadsheets
  - Compilers

Streams

- A flow of characters
- Input stream
  - Flow into program
    - Can come from keyboard
    - Can come from file
- Output stream
  - Flow out of program
    - Can go to screen
    - Can go to file

Streams Usage

- We've used streams already
  - cin
    - Input stream object connected to keyboard
  - cout
    - Output stream object connected to screen
- Can define other streams
  - To or from files
  - Used similarly as cin, cout

Streams Usage Like cin, cout

- Consider:
  - Given program defines stream inStream that comes from some file:
    int theNumber;
    inStream >> theNumber;
    - Reads value from stream, assigned to *theNumber*
  - Program defines stream outStream that goes to some file
    outStream << "theNumber is " << theNumber;
    - Writes value to stream, which goes to file

Files

- We'll use text files
- Reading from file
  - When program takes input
- Writing to file
  - When program sends output
- Start at beginning of file to end
  - Other methods available
  - We'll discuss this simple text file access here

File Connection

- Must first connect *file* to *stream object*
- For input:
  - File → ifstream object
- For output:
  - File → ofstream object

- Classes ifstream and ofstream
  - Defined in library <fstream>
  - Named in std namespace

File I/O Libraries

- To allow both file input and output in your program:

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

OR

```
#include <fstream>
using std::ifstream;
using std::ofstream;
```

Declaring Streams

- Stream must be declared like any other class variable:

```
ifstream inStream;
ofstream outStream;
```

- Must then "connect" to file:

```
inStream.open("infile.txt");
```

  - Called "opening the file"
  - Uses member function *open*
  - Can specify complete pathname

Streams Usage

- Once declared → use normally!

```
int oneNumber, anotherNumber;
inStream >> oneNumber >> anotherNumber;
```

- Output stream similar:

```
ofstream outStream;
outStream.open("outfile.txt");
outStream       << "oneNumber = " << oneNumber
                << " anotherNumber = "
                << anotherNumber;
```

  - Sends items to output file

File Names

- Programs and files
- Files have two names to our programs
  - External file name
    - Also called "physical file name"
    - Like "infile.txt"
    - Sometimes considered "real file name"
    - Used only once in program (to open)
  - Stream name
    - Also called "logical file name"
    - Program uses this name for all file activity

Closing Files

- Files should be closed
  - When program completed getting input or sending output
  - Disconnects stream from file
  - In action:

    inStream.close();
    outStream.close();
    - Note no arguments
- Files automatically close when program ends

File Flush

- Output often "buffered"
  - Temporarily stored before written to file
  - Written in "groups“
- Occasionally might need to force writing: outStream.flush();
  - Member function *flush*, for all output streams
  - All buffered output is physically written
- Closing file automatically calls flush()

File Example 1: Input/Output (1 of 2)

Display 12.1 Simple File Input/Output

```
1   //Reads three numbers from the file infile.txt, sums the numbers,
2   //and writes the sum to the file outfile.txt.
3   #include <fstream>
4   using std::ifstream;
5   using std::ofstream;
6   using std::endl;

7   int main( )
8   {
9       ifstream inStream;
10      ofstream outStream;

11      inStream.open("infile.txt");
12      outStream.open("outfile.txt");

13      int first, second, third;
14      inStream >> first >> second >> third;
15      outStream << "The sum of the first 3\n"
16                << "numbers in infile.txt\n"
17                << "is " << (first + second + third)
18                << endl;
```

A better version of this program is given in Display 12.3.

File Example 1: Input/Output (2 of 2)

```
19        inStream.close( );
20        outStream.close( );

21        return 0;
22    }
```

SAMPLE DIALOGUE

There is no output to the screen and no input from the keyboard.

infile.txt
(Not changed by program)

```
1
2
3
4
```

outfile.txt
(After program is run)

```
The sum of the first 3
numbers in infile.txt
is 6
```

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

Appending to a File

- Standard open operation begins with empty file
  - Even if file exists → contents lost

- Open for append:

```
ofstream outStream;
outStream.open("important.txt", ios::app);
```

  - If file doesn't exist → creates it
  - If file exists → appends to end
  - 2nd argument is class *ios* defined constant
    - In <iostream> library, std namespace

Alternative Syntax for File Opens

- Can specify filename at declaration

- ifstream inStream;
  inStream.open("infile.txt");

  EQUIVALENT TO:

  ifstream inStream("infile.txt");

Checking File Open Success

- File opens could fail
  - If input file doesn't exist
  - No write permissions to output file
- Member function fail()
  - Call fail() to check stream operation success/fail

```
ifstream gradeList("grades.txt");
if (gradeList.fail())
{
  cout << "File open failed.\n";
  exit(1);
}
```

- File stream object set to `0` (`false`) if open failed:

```
if (!gradeList) ...
```

Checking End of File

- Use loop to process file until end
  - Typical approach

- Two ways to test for end of file
  - Member function eof()

```
inStream.get(next);
while (!inStream.eof())
{
    cout << next;
    inStream.get(next);
}
```

    - Reads each character until file ends
    - eof() member function returns bool

End of File Check with Read

- Second method
  - read operation returns bool value!
    (inStream >> next)
    - Expression returns true if read successful
    - Returns false if attempt to read beyond end of file
  - In action:

```
double next, sum = 0;
while (inStream >> next)
  sum = sum + next;
cout << "the sum is " << sum << endl;
```

Tools: File Names as Input

- Stream open operation
  - Argument to open() is string type
  - Can be literal (used so far) or variable

```
char fileName[16];
ifstream inStream;
cout << "Enter file name: ";
cin >> fileName;
inStream.open(fileName);
```

  - Provides more flexibility

File Output Formatting

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

- Requires `iomanip` to use manipulators

Output Member Functions

- Consider:

```
outStream.setf(ios::fixed);
outStream.setf(ios::showpoint);
outStream.precision(2);
```

- Member function precision(x)
  - Decimals written with "x" digits after decimal

- Member function setf()
  - Allows multitude of output flags to be set

setf() Examples

- Common flag constants:
  - outStream.setf(ios::fixed);
    - Sets fixed-point notation (decimal)
  - outStream.setf(ios::showPoint)
    - Always include decimal point
  - outStream.setf(ios::right);
    - Sets right-justification

- Set multiple flags with one call:

```
outStream.setf(ios::fixed | ios::showpoint |  ios::right);
```

More Output Member Functions

- Consider:

  outStream.width(5);

- Member function width(x)
  - Sets width to "x" for outputted value
  - Only affects "next" value outputted
  - Must set width before each value in order to affect all
    - Typical to have "varying" widths
    - To form "columns"

Manipulators

- Manipulator defined:
  "A function called in nontraditional way"
  - Can have arguments
  - Placed after insertion operator
  - Do same things as member functions!
    - In different way
  - Common to use both "together“

- setw() and setprecision() are in library <iomanip>, std namespace

Manipulator Example: setw()

- setw() manipulator:
  cout << "Start" << setw(4) << 10
  << setw(4) << 20 << setw(6) << 30;
  - Results in:
    Start 10 20   30

- Note: setw() affects only NEXT outputted value
  - Must include setw() manipulator before each outputted item to affect all

Manipulator setprecision()

- setprecision() manipulator:

```
cout.setf(ios::fixed | ios::showpoint);
cout      << "$" << setprecision(2) << 10.3 << "  "
          << "$" << 20.5 << endl;
```

- Results in:
  $10.30 $20.50

File Example 2: Input/Output (1 of 3)

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 <cctype> // Needed for the toupper function.
6  using namespace std;
7
8  int main()
9  {
10    const int SIZE = 51;  // Array size for file name
11    char fileName[SIZE];  // To hold the file name
12    char ch;              // To hold a character
13    ifstream inFile;      // Input file
14
```

File Example 2: Input/Output (2 of 3)

Program 12-12 *(continued)*

```
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     inFile.open(fileName);
24     if (!inFile)
25     {
26        cout << "Cannot open " << fileName << endl;
27        return 0;
28     }
29
30     // Process the files.
31     inFile.get(ch);              // Get a char from file 1
32     while (!inFile.eof())        // Test for end of file
33     {
34        outFile.put(toupper(ch)); // Write uppercase char to file 2
35        inFile.get(ch);           // Get another char from file 1
36     }
37
38     // Close the files.
39     inFile.close();
40     outFile.close();
41     cout << "File conversion done.\n";
42     return 0;
43  }
```

File Example 2: Input/Output (3 of 3)

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?
```

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

File Example 3: Functions (1 of 3)

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  using namespace std;
6
7  // Maximum amount to read from a line in the file
8  const int MAX_LINE_SIZE = 81;
9
10 // Function prototypes
11 bool openFileIn(fstream &, char *);
12 void showContents(fstream &);
13
14 int main()
15 {
16    fstream dataFile;
17
18    if (!openFileIn(dataFile,"demofile.txt"))
19    {
20       cout << "File open error!" << endl;
21       return 0;    // Exit the program on error.
22    }
```

File Example 3: Functions (2 of 3)

Program 12-5 *(continued)*

```
23      cout << "File opened successfully.\n";
24      cout << "Now reading data from the file.\n\n";
25      showContents(dataFile);
26      dataFile.close();
27      cout << "\nDone.\n";
28      return 0;
29   }
30
31   //*************************************************************
32   // Definition of function openFileIn. Accepts a reference   *
33   // to an fstream object as its argument. The file is opened *
34   // for input. The function returns true upon success, false *
35   // upon failure.                                            *
36   //*************************************************************
37
38   bool openFileIn(fstream &file, char *name)
39   {
40      file.open(name, ios::in);
41      if (file.fail())
42         return false;
43      else
44         return true;
45   }
46
```

File Example 3: Functions (3 of 3)

```
47  //*************************************************************
48  // Definition of function showContents. Accepts an fstream  *
49  // reference as its argument. Uses a loop to read each name *
50  // from the file and displays it on the screen.             *
51  //*************************************************************
52
53  void showContents(fstream &file)
54  {
55     char line[MAX_LINE_SIZE];
56
57     while (file >> line)
58     {
59        cout << line << endl;
60     }
61  }
```

Program Screen Output

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

Jones
Smith
Willis
Davis

Done.
```

Summary

- Streams connect to files with open operation
- Member function fail() checks successes
- Stream member functions format output
  - e.g., width, setf, precision
  - Same usage for cout (screen) or files
- Stream types can be formal parameters
  - But must be call-by-reference