Queens College, CUNY, Department of Computer Science

Object Oriented Programming in C++

CSCI 211 / 611

Summer 2018

Instructor: Dr. Sateesh Mane

July 12, 2018

File I/O: reading/writing from/to file

- The material in this lecture is **not for examination.**
- This lecture explains how to **read and write from/to files** in C++.
- To do this, we employ **file streams (ifstream and ofstream).**
- An ofstream can be opened in different modes.
  1. "Overwrite mode" overwrites anything already in the output file.
  2. "Append mode" appends to the end of the output file.
  3. **The default is "overwrite" mode.**
- **If the output file does not exist, the ofstream will create it.**
- **If the input file does not exist, the ifstream file open will fail.**

1 Input file “infile.txt”

- In this example, the input file is named “infile.txt” and has three rows.
- Each row has an `int, double, string`.

```
6 2.2  Alice
2 1.6  Bob
5 3.4  Charlie
```

2 Example program

- Here is a working C++ program to read an input file and write to an output file.
- As a simple exercise, after reading in the data, we find the largest `int`, smallest `double` and longest `string`.
- We print the largest `int`, smallest `double` and longest `string` to an output file using “over-write” mode.
- Next we print the largest `int`, smallest `double` and longest `string` to the same output file using “append” mode.
- **We require the following headers for file I/O.**

```
#include <iostream>              // basic I/O
#include <fstream>               // file stream
```

- Our example also includes `<string>` and `<vector>`, but they are not connected with file I/O.

```
#include <iostream>
#include <fstream>                  // header for file I/O
#include <string>
#include <vector>

using namespace std;

int main()
{
  vector<int> iv;
  vector<double> dv;
  vector<string> sv;
  string infile("infile.txt");      // input file name
  string outfile("outfile.txt");    // output file name

  ifstream ifs(infile);             // ifstream object

  if (ifs.good() == false) {        // check if file opened successfully
    cout << "File not found: " << infile << endl;
    return 0;
  }

  while (true) {
    int a;
    double d;
    string s;
    ifs >> a >> d >> s;             // read from file
    if (ifs.eof() == true) break;   // end of file? break out of loop
    cout << a << "  " << d << "  " << s << endl;  // print for debugging
    iv.push_back(a);
    dv.push_back(d);
    sv.push_back(s);
  }

  ifs.close();  // optional, file will cloae automatically when ifs goes out of scope

  // find max int, min double, longest string
  int a_max = iv[0];
  double d_min = dv[0];
  string s_long = sv[0];
  for (int i = 0; i < iv.size(); ++i) {
    if (a_max < iv[i]) a_max = iv[i];
    if (d_min > dv[i]) d_min = dv[i];
    if (s_long.length() < sv[i].length()) s_long = sv[i];
  }
  ofstream ofs1(outfile, ios::out);  // "write mode": overwrites outfile
  ofs1 << a_max << endl;
  ofs1 << d_min << endl;             // write to file
  ofs1 << s_long << endl;
  ofs1.close();                      // close the file

  ofstream ofs2(outfile, ios::app);  // "append mode": add to end of file

  ofs2 << endl;
  ofs2 << "append mode: add to end of file" << endl;
  ofs2 << a_max << "   " << d_min << "   " << s_long << endl;

  ofs2.close();                      // close the file

  return 0;
}
```

3 Input stream, read from file

- We create an **ifstream** object.

  ifstream ifs(infile);

- Here “infile” is the name of the input file.
- **If the file does not exist, the file open procedure will fail.**

```
if (ifs.good() == false) {
  cout << "File not found: " << infile << endl;
  return 0;
}
```

- If the file open is successful, we loop and read the lines in the input file.
- We read the data into variables `int a`, `double d`, `string s`.
- When we reach end-of-file, we break out of the loop.

```
  ifs >> a >> d >> s;               // read from file
  if (ifs.eof() == true) break;    // end of file? break out of loop
```

- The syntax “`ifs >> (variable)`” is the same as “`cin >> (variable)`” to read from the console.
- In our example, we push back the values of a, d and s onto vectors.
- We close the file using the `close()` command.

```
ifs.close();  // optional, file will cloae automatically when ifs goes out of scope
```

- If we do not close the file, it will be closed automatically when `ifs` goes out of scope.

4 Max `int`, min `double`, longest `string`

- This is just an exercise to do something with the input data.
- It has no relevance to file I/O.

5 Output stream: overwrite mode

- We write some output to a file.
- We create an **ofstream** object.

  ofstream ofs1(outfile, ios::out);

- Here “outfile” is the name of the input file.
- Also “`ios::out`” states that the file is opened in “overwrite” mode.
- **If the file does not exist, the ofstream will create it.**
- Writing to file is similar to writing to the console.

```
ofs1 << a_max << endl;
ofs1 << d_min << endl;              // write to file
ofs1 << s_long << endl;
```

- We close the file using the `close()` command.

```
ofs1.close();                        // close the file
```

6 Output stream: append mode

- We create a different **ofstream** object.
- We use the same file name but now we use "append" mode.

  ofstream ofs2(outfile, ios::app);

- Here "outfile" is the name of the input file (same as before).
- Also "`ios::app`" states that the file is opened in "append" mode.
- When we write to the file, the output is added to the end of the file.
- **If the file does not exist, the ofstream will create it.**
- Writing to file is similar to writing to the console.

```
ofs2 << endl;
ofs2 << "append mode: add to end of file" << endl;
ofs2 << a_max << "   " << d_min << "   " << s_long << endl;
```

- We close the file using the `close()` command.

```
ofs2.close();                      // close the file
```

7 Output file “outfile.txt”

- Here is the output file for our example.

```
6
1.6
Charlie

append mode: add to end of file
6   1.6   Charlie
```