CIS 22A – Lecture 12

Manish Goel

Using Files

- Using files for program input, output allows data to be retained between program runs
  - *Open* the file
  - *Use* the file (read from, write to, or both)
  - *Close* the file

- Use `fstream` header file for file access

- File stream types:

  `ifstream` for input from a file
  `ofstream` for output to a file
  `fstream` for input from or output to a file

- Define file stream objects:

```
ifstream infile;
ofstream outfile;
```

Using Files - 2

- Create a link between file name (outside the program) and file stream object (inside the program)
- Use the `open` member function:

```
infile.open("inventory.dat");
outfile.open("c:\\docs\\report.txt");
```

- Filename may include drive, path info.
- Output file will be created if necessary; existing file will be erased first
- Input file must exist for `open` to work
- Use the `close` member function:

```
infile.close();
outfile.close();
```

- Don't wait for operating system to close files at program end:
  - may be limit on number of open files
  - may be buffered output data waiting to send to file

Filestreams ←→ I/O streams

- Can use output file object and << to send data to a file:

```
outfile << "Inventory report";
```

- Can use input file object and >> to copy data from file to variables:

```
infile >> partNum;
infile >> qtyInStock >> qtyOnOrder;
```

- Correspondence to I/O streams

```
#include <iostream>   ←→   #include <fstream >
cin >>                ←→   ifstream infile >>
cout <<               ←→   ofstream outfile <<
```

Specifying Filenames

- `open` function requires name of the file as a null-terminated string, also known as a *C-string*.
- *String literals are stored* in memory as null-terminated C-strings, but *string objects* are not.
- `string` objects have a member function named `c_str` which returns the contents of the object formatted as a null-terminated C-string.

```
ofstream outfile;
string myStr = "abc.txt";
outfile.open(myStr.c_str());
```

File Open Errors and Loops

- Test a file stream object to detect if an open operation failed:

```
infile.open("test.txt");
if (!infile)
{
  cout << "File open failure!";
}
```

- Can also use the `fail` member function `infile.fail()`

- Stream extraction operator `>>` returns `true` when a value was successfully read, `false` otherwise

- Can be tested in a `while` loop to continue execution as long as values are read from the file:

```
while (inputFile >> number) ...
```