

Goals for This Lecture:

- Understand I/O to/from files in C++
- Introduce least-squares techniques for fitting models to data

C++ File I/O

- **C++ provides objects that allow I/O to and from files**
- **Instances of these objects are called streams**
- **We have already made use of the STDOUT/STDIN streams cout & cin**
- **We show how to use these file I/O streams by example**

Reading Values from a File

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ifstream inStream;           // Declare the input stream
    double x, y;                 // Declare some variables

    inStream.open("input.dat");  // Open the file for reading

    inStream >> x >> y ;        // Read x & y from the file

    cout << " x, y = " << x << " " << y << endl;

    inStream.close();           // Close the file

    return(0);                  // Return a no-error value
}
```

C++ ifstream Objects

- C++ ifstream objects have a number of important methods that can be used to control input from the stream

- Open:

```
ifstream.open(file_name_string);
```

- Opens a file stream for reading

- Close:

```
ifstream.close();
```

- Closes a file stream

- Fail:

```
ifstream.fail();
```

- Indicates a failure on the input stream
 - A file open operation failed
 - Extraction of some values from the stream failed

- Clear:

```
ifstream.clear();
```

- Clears the stream for future reuse; should be called after closing file

Reading an Arbitrary Amount of Data from a File, pt. 1

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    ifstream inStream;           // Declare the input stream
    double x, y;                 // Declare some variables
    double *xarray, *yarray;    // Declare dynamic arrays
    int npnts;                   // Number of points

    inStream.open("input.dat");  // Open the file for reading
    while(!inStream.fail()) {
        inStream >> x >> y ;    // Read x & y from the file
        if(inStream.fail()) break; // If a failure exit the loop
        npnts = npnts+1 ;       // Increment the data counter
    }
    inStream.close();           // Close the file
    inStream.clear();           // Clear the stream for reuse
```

Reading an Arbitrary Amount of Data from a File, pt. 2

```
xarray = new double[npnts] ; // Create the x-array
yarray = new double[npnts] ; // Create the y-array

inStream.open("input.dat"); // Open the file for reading
    for(int i=0; i < npnts ; i++)
        inStream >> xarray[i] >> yarray[i] ; // Read in data
inStream.close(); // Close the file
inStream.clear(); // Clear the stream for reuse

return(0); // Return a zero error code
}
```

Assignment

- Read Savitch Sections 12.1 & 12.2.