

Reading and Writing Files

Exercises

How to open a file for reading

- Describe how to open a file for reading (i.e., receiving input from the file)
- How can we tell whether the file was successfully opened?

- (No exercises for this slide)

- Write some sample code which opens a file and reads text from it, one word at a time
- Are there any disadvantages to reading it this way?
- Convert your code to a full program (remember to include `fstream`). You will need to put the input text file in the same directory that the program will be executed in
 - For an IDE, this can usually be done by creating a new file in the project

Reading from a file (2)

- Write some sample code which opens a file and reads text from it, one line at a time, into a `std::string` variable
- What data will be stored in the variable after each line is read?
- Convert your code to a full program as described in the previous slide

- Convert this sample code to a full working program. Call the output file `hello_out.txt` to avoid overwriting the other file
- Verify that the output file `hello_out.txt` has been created and contains the correct text.
 - The file will usually be in the same directory that the program runs in. If you are using an IDE, you may need to check the project settings to find where this is

- (Revision) Explain what is meant by flushing a stream
- iostreams are usually flushed at the end of each line of input. When is an fstream usually flushed and why is this different from iostreams?
- Are there any advantages and disadvantages to this approach?
- Give an example of where this approach could cause problems. How can you avoid this?

Closing files

- When we call `close()` on an `ofstream` and there is still data in the buffer, what happens to it?
- What happens if we forget to call `close()` on an `fstream`?