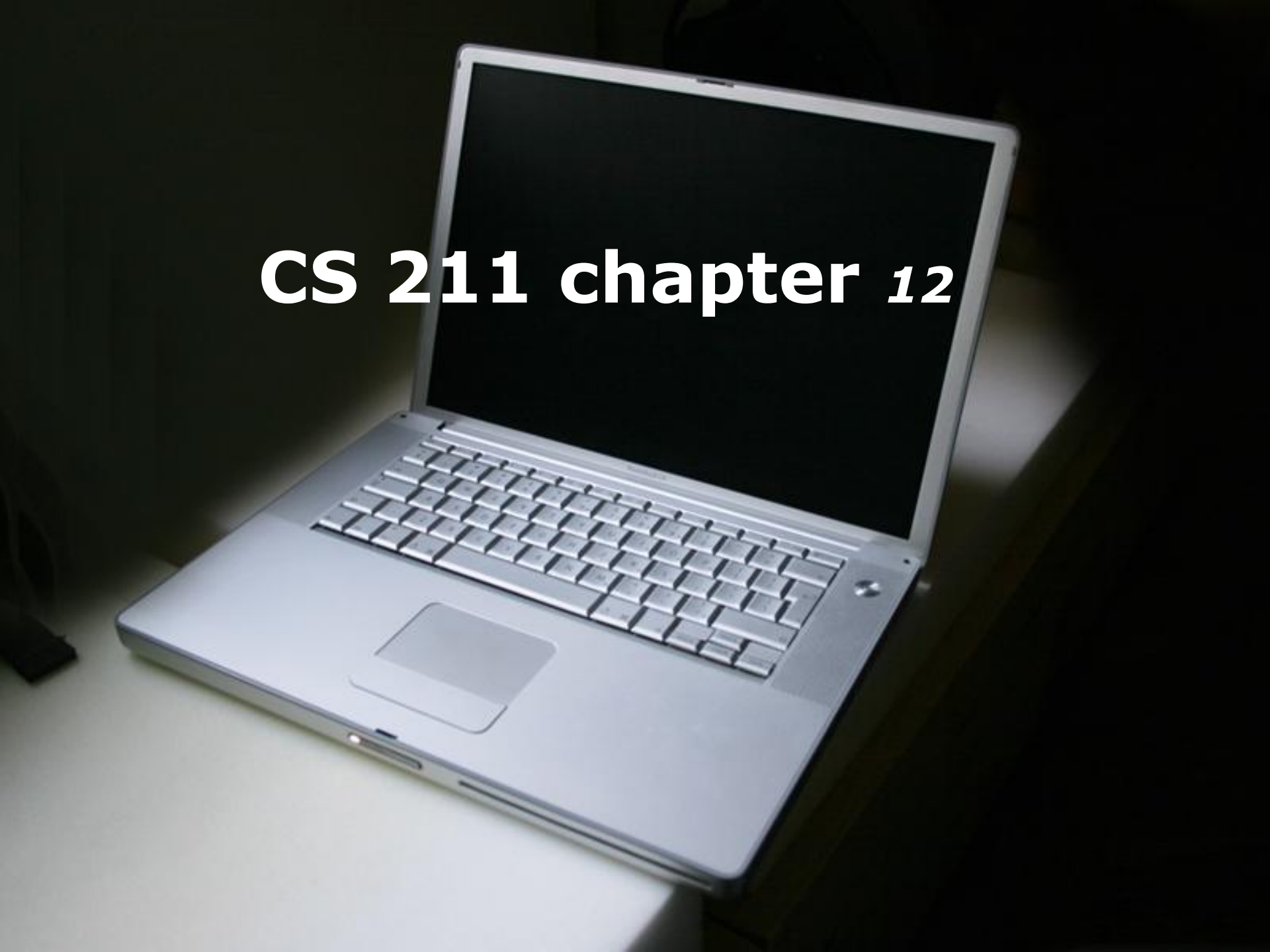

CS 211 chapter 12

Input Stream:

- Currently we get all the input from cin which is entered by keyboard.

- Every input stream has member function get().

Files:

- Currently when we work with arrays, each time we need the data from user, they have input by hand.

- Instead of re-enter it every time, we can just save the data in a file and the program can access the file, retrieve it from there.

- It's more user friendly and more efficient.

Setup:

- C++ has a library setup specifically for file access.

- To use this in our program we need to include, *fstream*, file stream library
  #include <fstream>

- In our main program we now have access to data type call fstream.

Fstream Variable:

- We will need a fstream variable to help us keep track of the file, usually one file per variable.

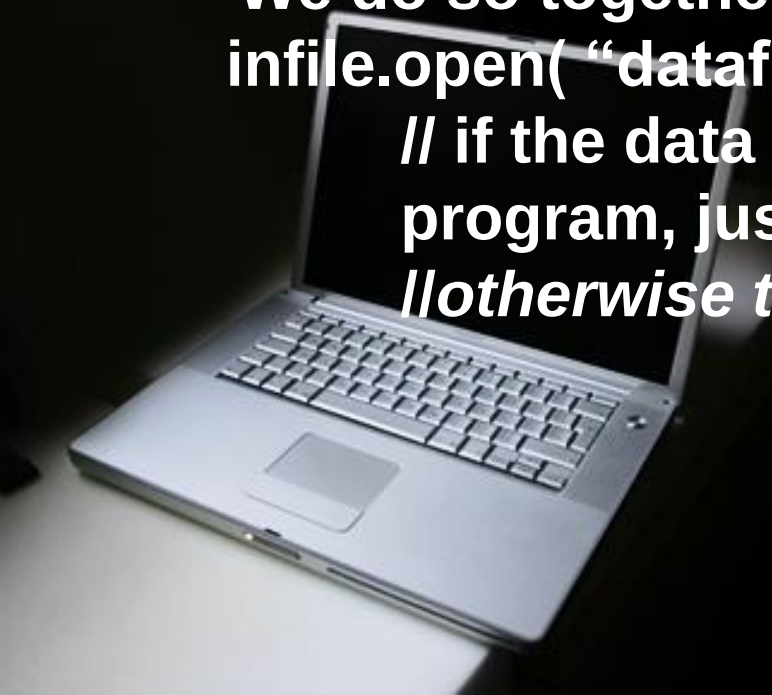
- Assume, we want to read a file, let's call it *infile*.
  ifstream infile;

Open file for reading:

- To gain access to the file, we need to specify two pieces of information:
  - Name of the file, maybe path to the file
  - How we are going to use the file

- We do so together with an open function call:

```
infile.open( "datafile.txt" );
        // if the data is in the same place as this
        program, just provide the full name of the file
        //otherwise the path to the file
```



Check Whether Opened Successfully:

- After we make the function call REQUEST to open the file for reading, we can't assume it worked successfully, what if the file doesn't exist?

- Here is how we check it:

```
If ( ! infile.is_open() ) {
   cerr << "File failed to open." << endl;
   return;
}
```

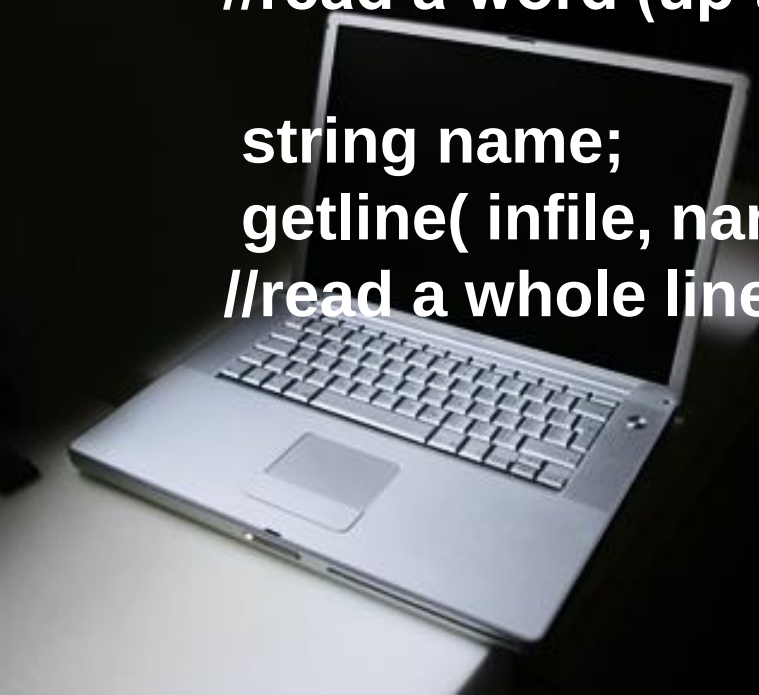
Read from the file:

Once the file is properly opened for access to read, we can read *fstream* the same way we do for *cin*.

Ex:

```
 int num;
 infile >> num;
//read a word (up to space, but won't eat up the space)

 string name;
 getline( infile, name);
//read a whole line (up to the end of the line \n)
```

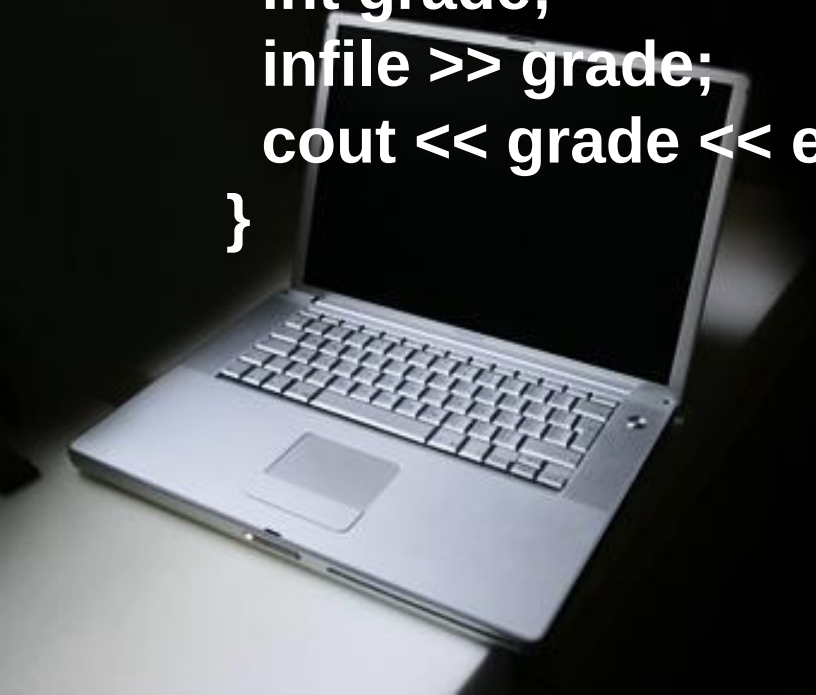


Read all from the file:

- If the file is very long, we can use a while loop to read everything. But where to stop?

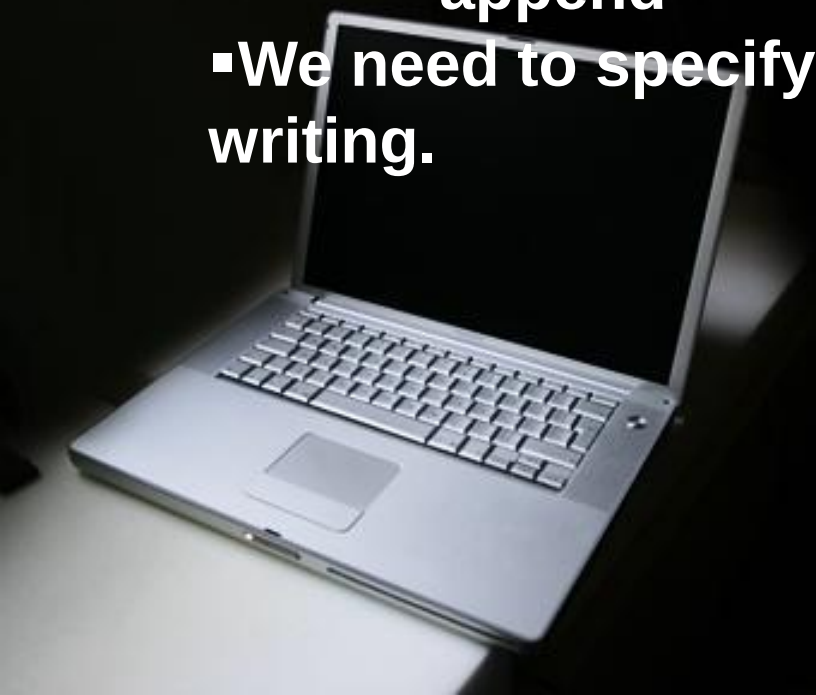
- Read until the end of file – eof :

```
while( ! Infile.eof() )
{
  int grade;
  infile >> grade;
  cout << grade << endl;
}
```



Write to a file:

- Again, we need a fstream variable, then open it to serve as output.
- Writing to a file, there're two options:
  - Erase(overwrite) the existing content on the file
    - truncate
  - Add it to the end of existing content
    - append
- We need to specify it when opening the file for writing.

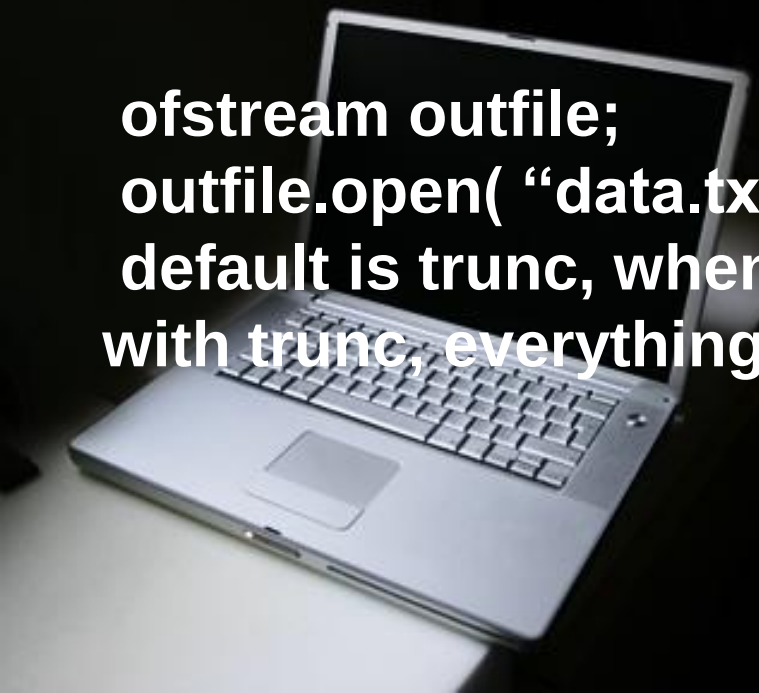


Open for Overwrite:

```
fstream outfile;
outfile.open( “data.txt”, fstream::out | ios::trunc );
```

Open for Append:

```
fstream outfile;
outfile.open( “data.txt”, fstream::out | ios::app );
```



```
ofstream outfile;
outfile.open( “data.txt” );
```

default is trunc, whenever the open() function is called with trunc, everything in the original file will be erased.

Writing to file:

- Once open call has been made, you may perform the same check.

- Write to the file is exactly the same as you would for cout.

```
outfile << "This sentence is going to data.txt.";
```

Text might store in the buffer and not write onto the file immediately, to force it and make sure it's written to the file: `outfile.flush();`

Once again, after you done access it, you may close the file: `outfile.close();`

It automatically flush everything from the buffer when close the file.