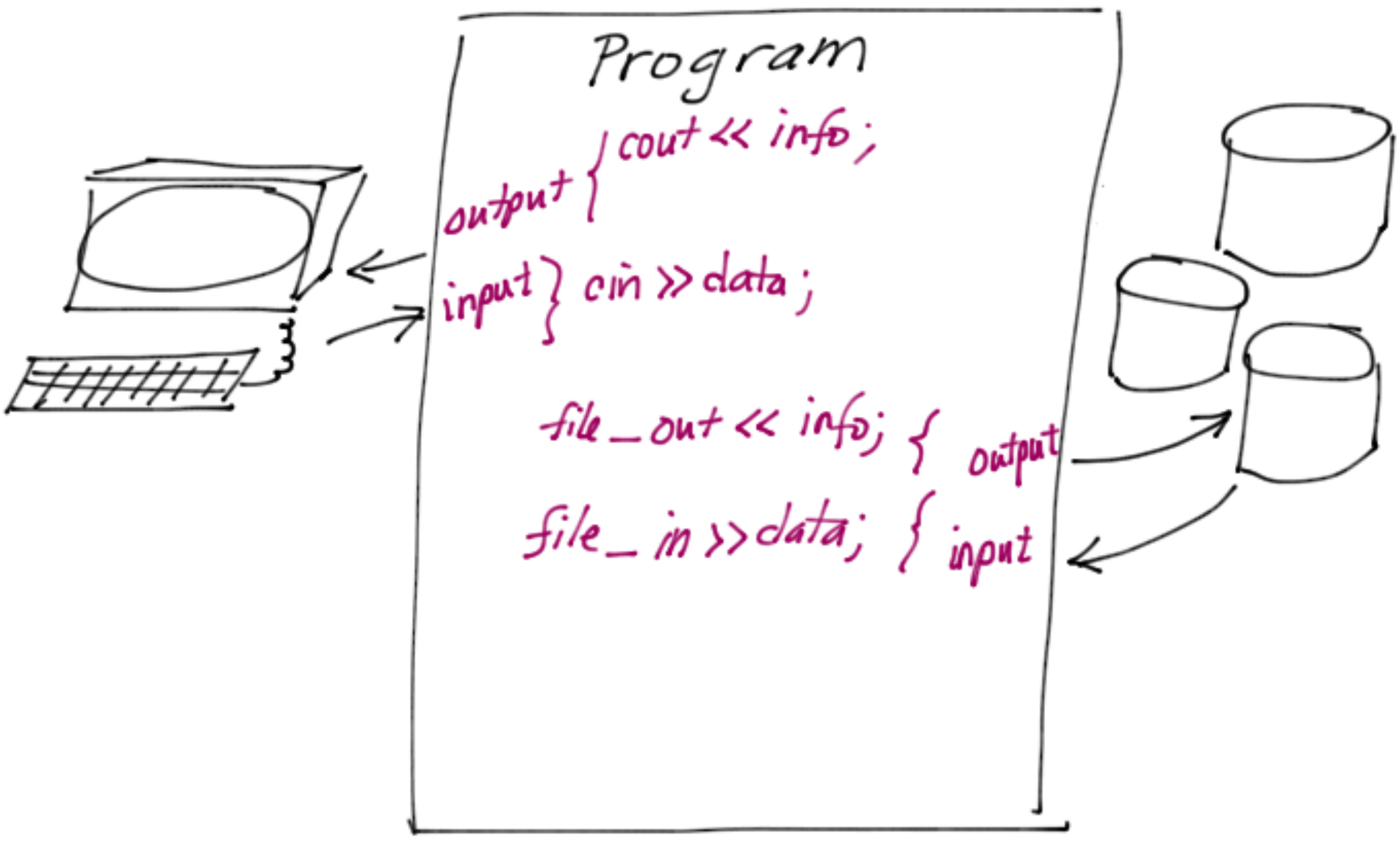
Lecture 8 - CS162

1. External Data Files
   - output to a file
   - input from a file

2. Writing programs with structs <u>and</u> external data files

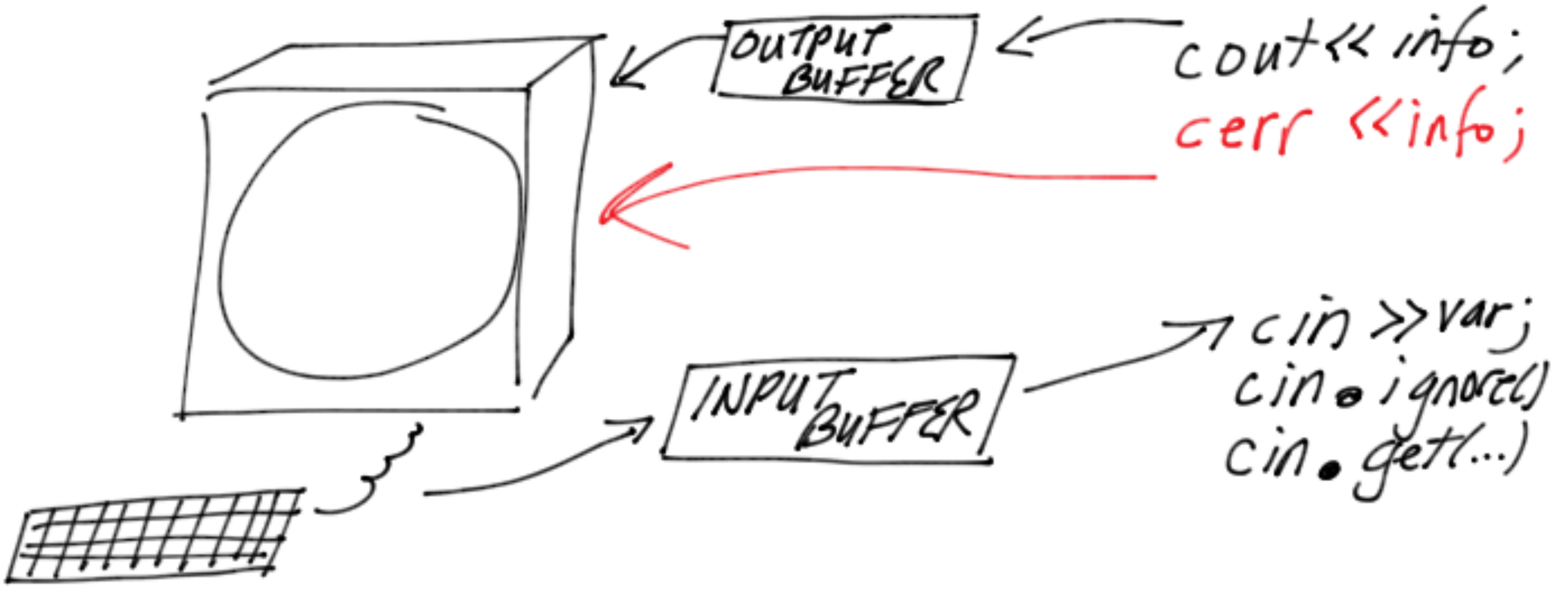
This lecture assumes we are working with $\boxed{\text{Text}}$ files (human readable material)



* Everything we already know about I/O works the same when working with files, except instead of cin & cout we use file variables connected to the desired file.

Understanding I/O

```
ostream cout; }  global variables defined
istream cin;  }  in the iostream library
```



Applying this to Files

```
#include <fstream>
```
← allows us to work with files

//Next we need file variables (because cin is tied to standard-in and cout is tied to standard-out, and we wouldn't want to change this.

```
ifstream file_in;
ofstream file_out;
```
← FOR INPUT FROM a File

FOR OUTPUT $\boxed{\text{TO}}$ a file

// But these variables are not yet connected to any files.

Writing $\boxed{\text{OUT}}$ to a file:

```
#include <fstream>
#include <iostream>
using namespace std;
```

} make sure to do this

```
//inside a function

ofstream file_out;          //set up a file variable

file_out.open("filename.extension");   //connects to a
                                          file

file_out << name <<endl;
```

Examples

```
file_out.open("inv.dat");
```

vs

```
char filename[31];
cin >>filename;
cin.ignore(100, '\n');
file_out.open(filename);
```

$\underbrace{\text{filename}}_{\text{an array of characters}}$

Important

1. when you open a file for output the contents of the file is $\boxed{\text{LOST}}$

2. The code from the previous page will be written at the BEGINNING of the file

3. IF you want to preserve the information that was in the file, then open the file for $\boxed{\text{APPEND}}$

```
file_out.open(filename, ios::app);
```

a Literal string or array of characters

append mode

4. Now new information is written after the last item in the file. Make Sure to write a newline or other delimiter so that we can distinguish between the data

Reading $\boxed{\text{IN}}$ from a file

```
#include <fstream>   // etc.
```

Inside a function:

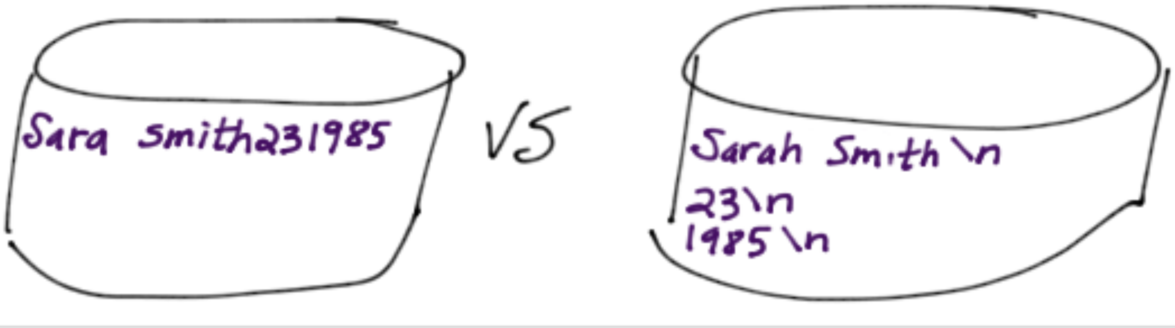
```
    ifstream file_in;  // Be careful of the name used

  file_in.open(filename);
```

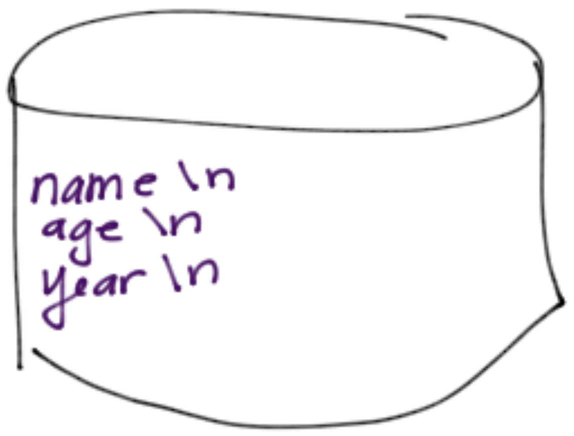
① Always begins reading at the beginning of the file

② Make sure there are <u>delimiters</u> in the file between fields

** there needs to be a way to read information back from the file



Everything we know applies!



```
file_in.get(namearray, size, '\n');
file_in.ignore(100, '\n');

file_in >> age;
file_in.ignore();

file_in >> year;
file_in.ignore();
```

But, when does input end?

— we can't prompt the "file"!!

∴ When we try to read from a file and there is nothing there, end of file (a "state" variable in the fstream Library) gets set.

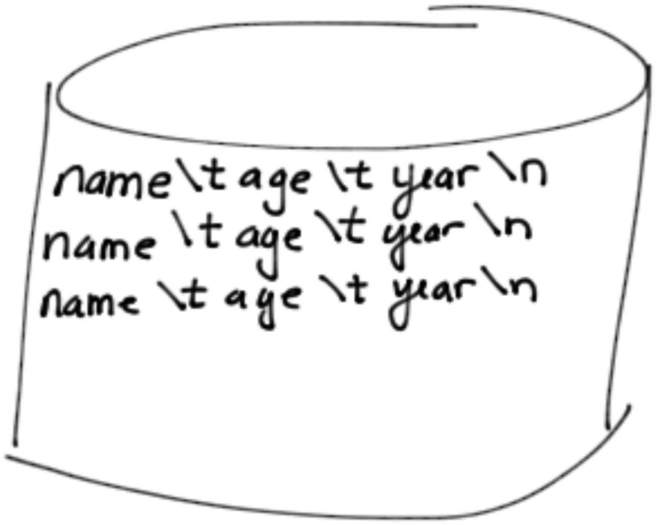
Detecting End of File

1) We **must** attempt to **read** from the file to find out if there is anything in the file to read

2) $\underbrace{\text{file\_in}}\,.\,\underbrace{\text{eof}}_{\leftarrow \text{function call}}()$

   - Returns **true** if the previous read/input operation failed
   - Returns **false** if the previous read/input was successful
   - Therefore, $\boxed{\text{BEFORE}}$ you check end of file, make sure to attempt to read $\boxed{\text{FIRST}}$. "Prime the Pump"

Let's Read from a file!



```
ifstream fin;  //file variable
fin.open(filename);
if (fin)  //we are connected
{
   fin.get(name, size, '\t');
   fin.ignore(100, '\t');   <- notice
   while (!fin.eof())
   {  //we are not yet at end
      fin >> age; fin.ignore();
      fin >> year;
      fin.ignore(100, '\n');

      //Now prime the pump.....
      // is there another?
      fin.get(anothername, size, '\t');
      fin.ignore(100, '\t');
   }
```

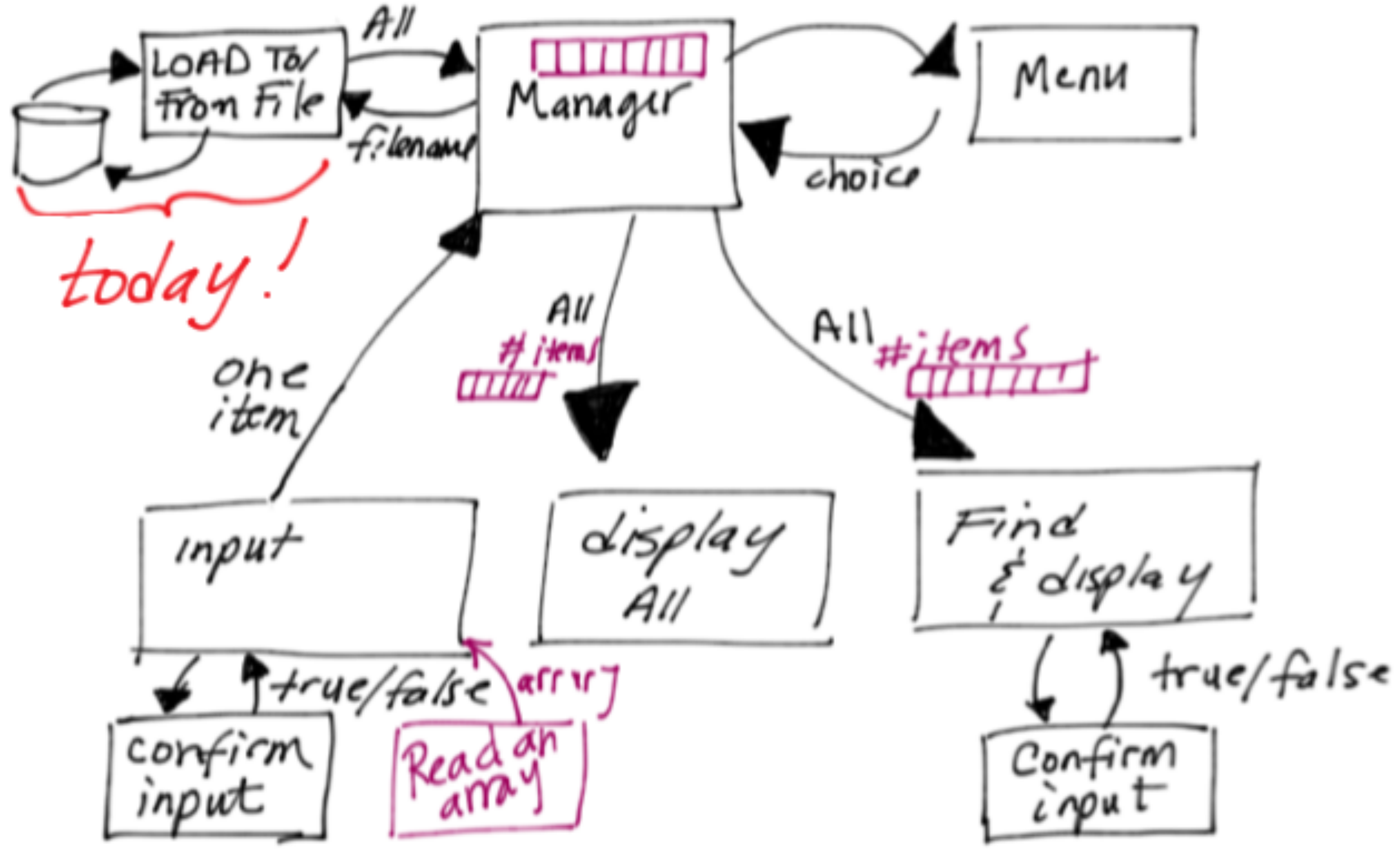
we have more data

handle the data.... display.....

is there another or are we done?

Adding External Files
– to our design

- Although a struct allows us to group different kinds of data – there are no operations built-in w/ structs besides memberwise copy (=)



LOAD To/ From File
All
filename
Manager
Menu
choice
today!
one item
All
# items
All
# items
input
display All
Find & display
confirm input
true/false
array
Read an array
true/false
Confirm input