CS 103 Unit 4f – File I/O and File Streams

Getting data into and out of files stored on disk using `ifstream` and `ofstream` objects

SIMPLE FILE I/O USING FILESTREAM OBJECTS

getline() Member Function and Lines of Text

- `cin` stops reading at whitespace
  - If you want to read a whole line of text use **`cin.getline()`**
    - It will read spaces and tabs but STOPS at '\n'
  - `cin.getline(char *buffer, int max_chars)`
    - Reads `max_chars-1` leaving space for the null character

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

int main ()
{
  char mytext[80];
  cout << "Enter your full name" << endl;
  cin.getline(mytext, 80);
  // vs. cin >> mytext;

  int last=0;
  for(int i=0; i<80; i++){
    if(mytext[i] == ' '){
       last = i+1;
       break;
    }
  }
  cout << "Last name starts at index: ";
  cout << last << endl;
  return 0;
 }
```

```
Enter your full name
Tommy Trojan
Last name starts at index 6.
```

Input Stream Error Checking

- We can check errors when **cin** receives unexpected data that can't be converted to the given type
- Use the function fail() member function (i.e. cin.fail()) which returns true if anything went wrong opening or reading data in from the file (will continue to return true from then on until you perform cin.clear())
- For now (in CS103), print an error and exit.

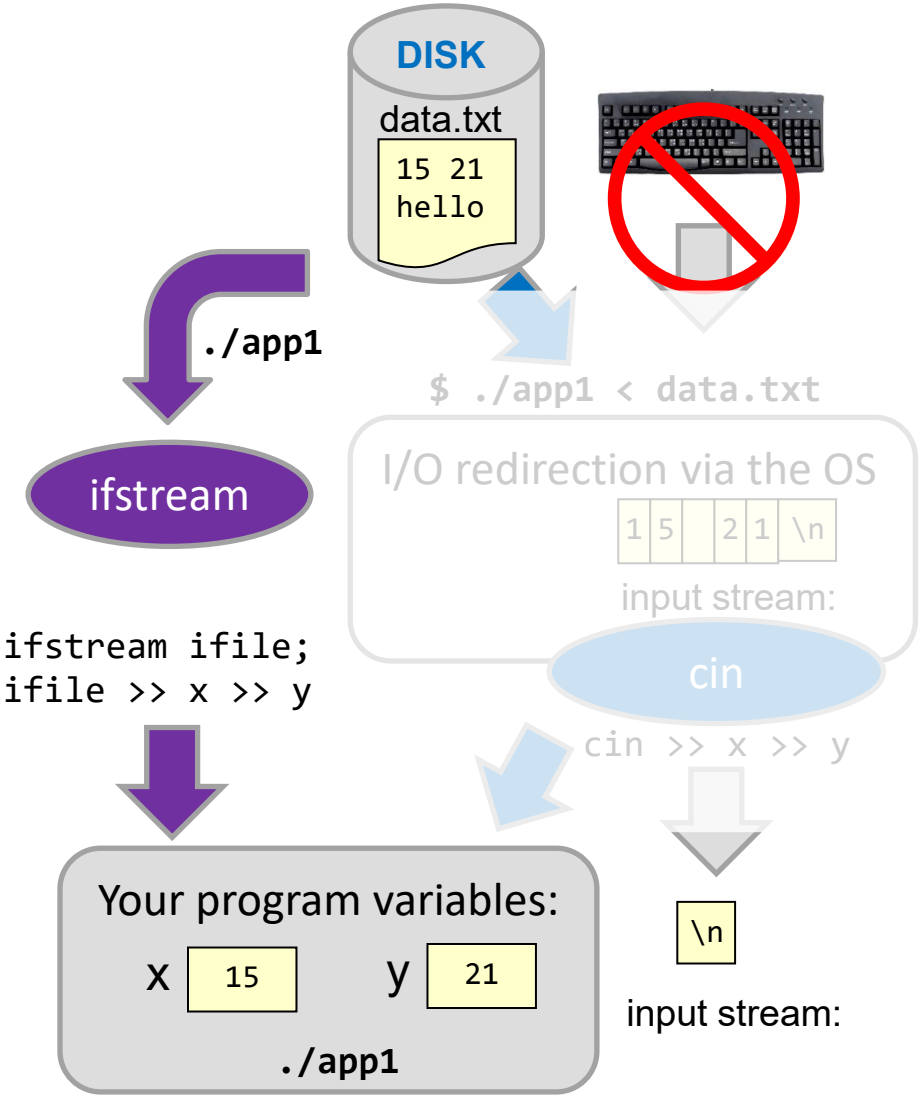
```
#include <iostream>
using namespace std;

int main ()
{
  int x, y;
  cout << "Enter an int: " << endl;

  cin >> x;  // What if the user enters:
             //    "abc"

  // Check if we successfully read an int
  if(  cin.fail() ) {
    cout << "Error:  I said enter an int!";
    cout << " Now I must exit!" << endl;
    return 1;
  } else {
    cout << "You did it! You entered an int";
    cout << " with value: " << x << endl;
  }

  cin >> y; // Will fail regardless
  if(  cin.fail() ) { ... } // always true
  return 0;
}
```

File I/O Intro

- What methods does a user have to provide a program input:
  - cin
  - Command line (argc, argv)
- Now a **third** method: **File I/O** (accessing data in files)
- Primary method for a program to read/write files:
  - **File streams [Main subject of our lecture]**
  - I/O Redirection [Subject of a later lecture] which uses the OS to "redirect" input and output from cin and cout to files instead of the keyboard and monitor.

Overview

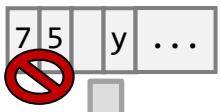
- Two methods for file I/O
  - **File streams** (ifstream and ofstream) are part of the C++ library and perform file I/O directly through a cin- and cout-like interface
  - **I/O redirection**: [More on this later]
    - The OS reads or writes data to/from a file by controlling cin & cout
    - The program just performs normal cin and cout commands



Recall: I/O Streams

- C++ and the OS use the notion of **streams** to temporarily store (aka buffer) data to be input or output and then uses the `cin` and `cout` **objects** (from the `<iostream>` library) to access those streams. (These objects have internal data members and member functions).
- While cin has a stream that receives input from the keyboard and cout has a stream that outputs to the terminal, the same stream **abstraction** could be applied to **files**!

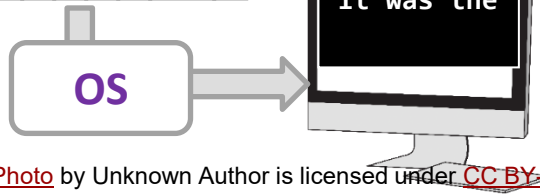
input stream
memory (aka stdin):




```
#include<iostream>
int main()
{
  int x;
  std::cin >> x;
  return 0;
}
```

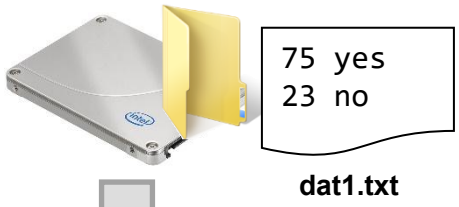
```
#include<iostream>
int main()
{
  std::cout << "It was the" << std::endl;
  std::cout << "best of times.";
  return 0;
}
```

output stream
memory (aka stdout):



File Streams

- C++ leverages the SAME interface that cin and cout provide to:
  - Read data IN from a file (like `cin`, but data comes from a **file** not the keyboard) and
  - Write data OUT to a file (like `cout`, but data goes to a **file** not the terminal).
- The counterpart to `cin` is an `ifstream` object
- The counterpart to `cout` is an `ofstream` object

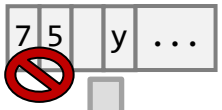


```
75 yes
23 no
```

dat1.txt

OS

input stream
memory:



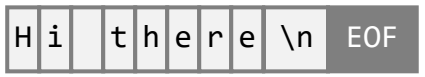
ifstream

```
#include <iostream>
#include <fstream>
Using namespace std;
int main() {
  int x;
  ifstream ifile("dat1.txt");
  ifile >> x;
  // use x
  ifile.close();
  return 0;
}
```

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
  ofstream ofile("dat2.txt")
  ofile << "Hi there" << std::endl;
  ofile.close();
  return 0;
}
```

ofstream

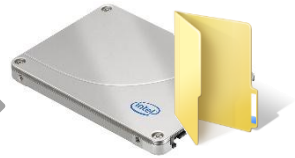
output stream
memory:



OS

```
Hi there
```

dat2.txt

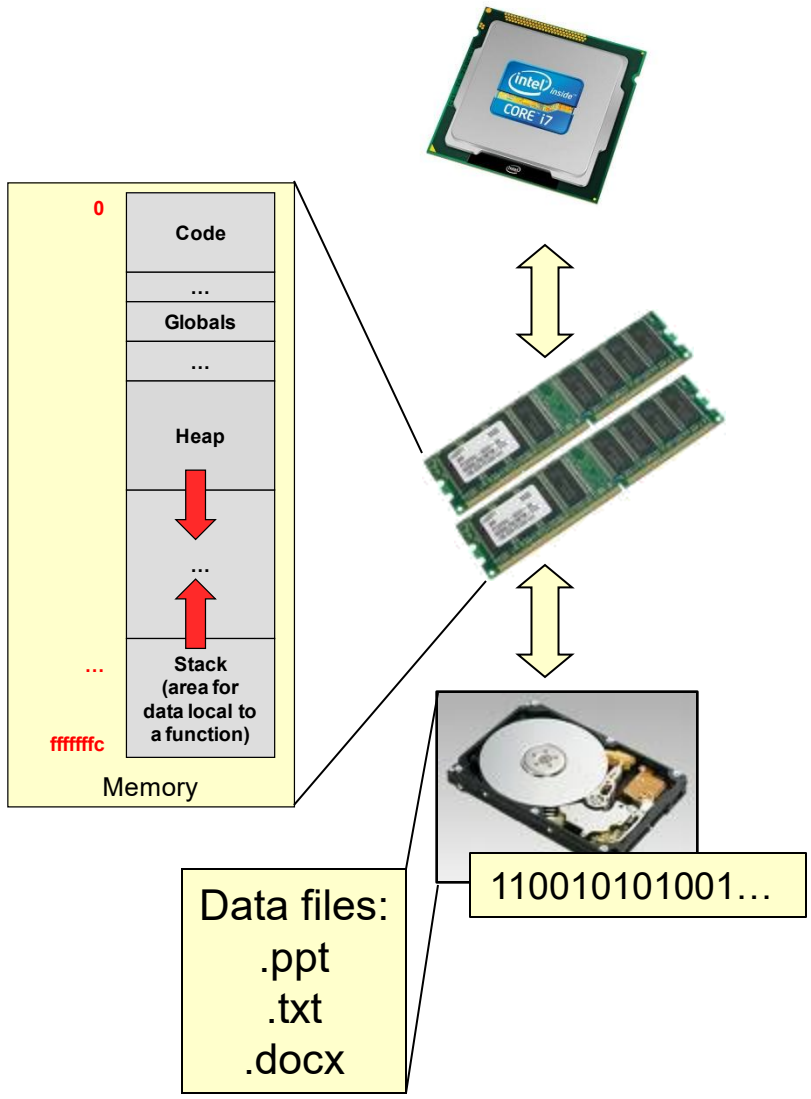


Input & Output Streams

- To summarize, there are other types of input and output streams other than **`cin`** and **`cout`**
- File streams gives the same capabilities as **`cin`** and **`cout`**, except data is read/written from/to a file on the hard drive
  - Everything you do with **`cin`** using the '>>' operator you can now use with an **`ifstream`** to access data from a file
  - Everything you do with **`cout`** using the '<<' operator you can now use with an **`ofstream`** to output data to a file
- Let's learn more about streams '>>'…we'll see it in the context of **`cin`** and **`cout`** but realize it will apply to other streams we'll learn about next

Disk Access

- All code and data (variables) resides in RAM
- Processor can ONLY talk directly to RAM (memory)
  - Cannot access files on disk directly
- Files on your disk require special OS routines to access their data
  - The OS provides routines to perform the translation
- C++ provide file streams to abstract those OS routines and help to
  - Take data from a file and input it to your variables or
  - Take your variables and writes their values to a file



Important Fact

- For your program to operate on data in a file…
- …you must read it into a variable
- Everything we will see subsequently is simply how to
  - Get data from a file into a variable, or
  - Take data from a variable and save it to a file

Two Kinds of Files: Binary and Text

- We conceive of files as "streams" (1D arrays) of data
- Files are broken into two types based on how they represent the given information:
  - Text files: File is just a large sequence of ASCII characters (every piece of data is just a byte)
  - Binary files: Data in the file is just bits that can be retrieved in different size chunks (4-byte int, 8-byte double, etc.)
- Example: Store the number 172 in a file:
  - Text: Would store 3 ASCII chars '1','7','2' (ASCII 49, 55, 50) requiring 3 bytes (but the key is each digit or character is stored separately in ASCII)
  - Binary: If 172 was in an 'int' variable we could store it as 4-bytes in binary rather than separate ASCII characters.

In this unit, we will only focus on text file I/O

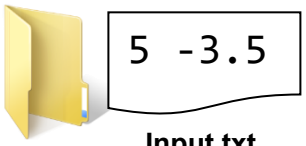
Starting File I/O

- Just like document or image editor allows two operations on files:
  - Read info from the file (like 'Open' command)
    - Use an `'ifstream'` object to open the file
    - Read data from the file
    - Close it when you're done
  - Write info to the file (like 'Save As' command)
    - Use an `'ofstream'` object
    - Write the data to a file
    - Close it when you're done

TEXT FILE I/O

Text File I/O - ifstream

- Must include <fstream>
- Use **ifstream** object/variable for reading a file
  - Can do anything 'cin' can do
  - Must "open" the file (usually by specifying the filename when you create the ifstream
- Use '>>' operator, **.fail(), .getline()** with the ifstream object just as you would on cin but realize operations are happening on data from the file




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

int main ()

{

  int x; double y;

  ifstream ifile ("input.txt");

  if(  ifile.fail() ){  // able to open file?
    cout << "Couldn't open file" << endl;
    return 1;
  }

  ifile >> x >> y;
  if (  ifile.fail() ){
    cout << "Unable to read int & double." << endl;
    return 1;
  }

  //...









  return 0;
}
```

Text File I/O - ofstream

- Must include <fstream>
- Use ofstream object/variable for writing to a file
  - Can do anything 'cout' can do
- Use '>>' operator, I/O manipulators, etc. on the stream but realize operations are happening on data form the file
- Close all filestreams using the .close() member function

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

int main ()

{

  int x; double y;

  ifstream ifile ("input.txt");

  if(  ifile.fail() ){  // able to open file?
    cout << "Couldn't open file" << endl;
    return 1;
  }

  ifile >> x >> y;
  if (  ifile.fail() ){
    cout << "Unable to read int & double." << endl;
    return 1;
  }
  ofstream ofile("output.txt");

  ofile << "Int from file is " << x << endl;
  ofile << "Double from file is " << y << endl;


  ifile.close();
  ofile.close();

  return 0;
 }
```

```
5 -3.5
```

Input.txt

```
Int from file is 5
Double from file is -3.5
```

output.txt

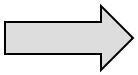
Getting Lines of Text

- Recall, using the >> operator to get an input string of text (char * or char [] variable passed to cin) ***implicitly stops at the first whitespace***
- To get a whole line of text (including spaces)
  - `cin.getline(char *buf, int bufsize);`
  - `ifile.getline(char *buf, int bufsize);`
  - Reads max of bufsize-1 characters (including newline)
    - Reads and discards the newline and places NULL char at the end of the string.
- This program reads all the lines of text from a file and adds "line numbers".

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

int main ()

{

  char myline[100]; int i = 1;

  ifstream ifile("input.txt");

  if(  ifile.fail() ){  // can't open?
    return 1;
  }

  ifile.getline(myline, 100);
  while ( ! ifile.fail()) {
   cout << i++ << ": " << myline << endl;
   ifile.getline(myline, 100);
  }

  ifile.close();
  return 0;
}
```

```
The fox jumped over the log.\n
The bear ate some honey.\n
The CS student solved a hard problem.\n
```

input.txt



```
1: The fox jumped over the log.
2: The bear ate some honey.
3: The CS student solved a hard problem.
```

Activity

- File I/O Exercises