CSCI 1300

File I/O, Classes
June 30th, 2021

University of Colorado **Boulder**

Be Boulder.

Please use the github link for the programing examples and slides.

https://github.com/rahul-aedula95/CSCI-1300

File I/O

- Taking input from files is more common than taking it from the terminal.
- Most scenarios in computer science requires processing data which is often stored in files (in different formats).
- Learning to read files (with their formatting) and writing results to files is really important.
- There are many different ways to do this, we’ll discuss the general ones.

Reading a file

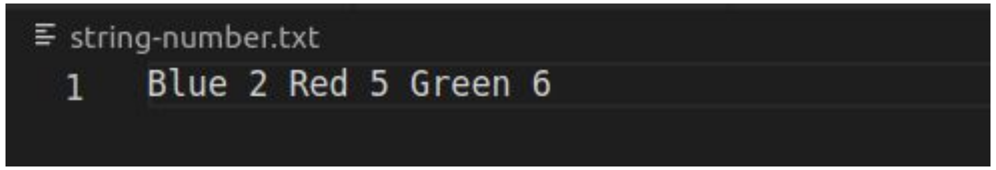
Streams are the concept c++ input and output uses.

Refer to example1.cpp in Week 5 of our github page

- To input a file we need to make an input stream.
- ifstream - this is used to create a plain text input stream
- syntax - ifstream fp;
- here fp is file pointer. This is just a variable which is referencing that it will be a file
- you may also declare it as: ifstream fp(“filename.txt);
- remember delimiters are characters that separate data in these files
- Here blank spaces are delimiters.
- This reads the file word by word

Reading a file

- Refer to example3.cpp in Week 5 of our github page
- You may also read different types words with preset type.
- Let is take an example of a file



string-number.txt

```
Blue 2 Red 5 Green 6
```

- Here the words are have different data types, string and integer.
- You may read them as strings and convert them into int using string conversion such as stoi
- You can also use the >> operator as demonstrated in the example.

Reading a file line by line

- Refer to example4.cpp and example5.cpp in Week 5 of our github page
- We can use getline (we used this command back when we were taking in strings) to read each line of the file directly into a string.
- Remember getline takes string only, so if you input a data type which needs to be used like int, float etc you need to convert it.
- You may also use getline to take in tabular input as demonstrated in example5.cpp

Writing a file

Refer to example2.cpp in Week 5 of our github page

- To output a file we need to create an output stream
- ofstream - this is used to create a plain text output stream
- syntax - ofstream fp;
- you may also declare it as: ofstream fp("filename.txt);

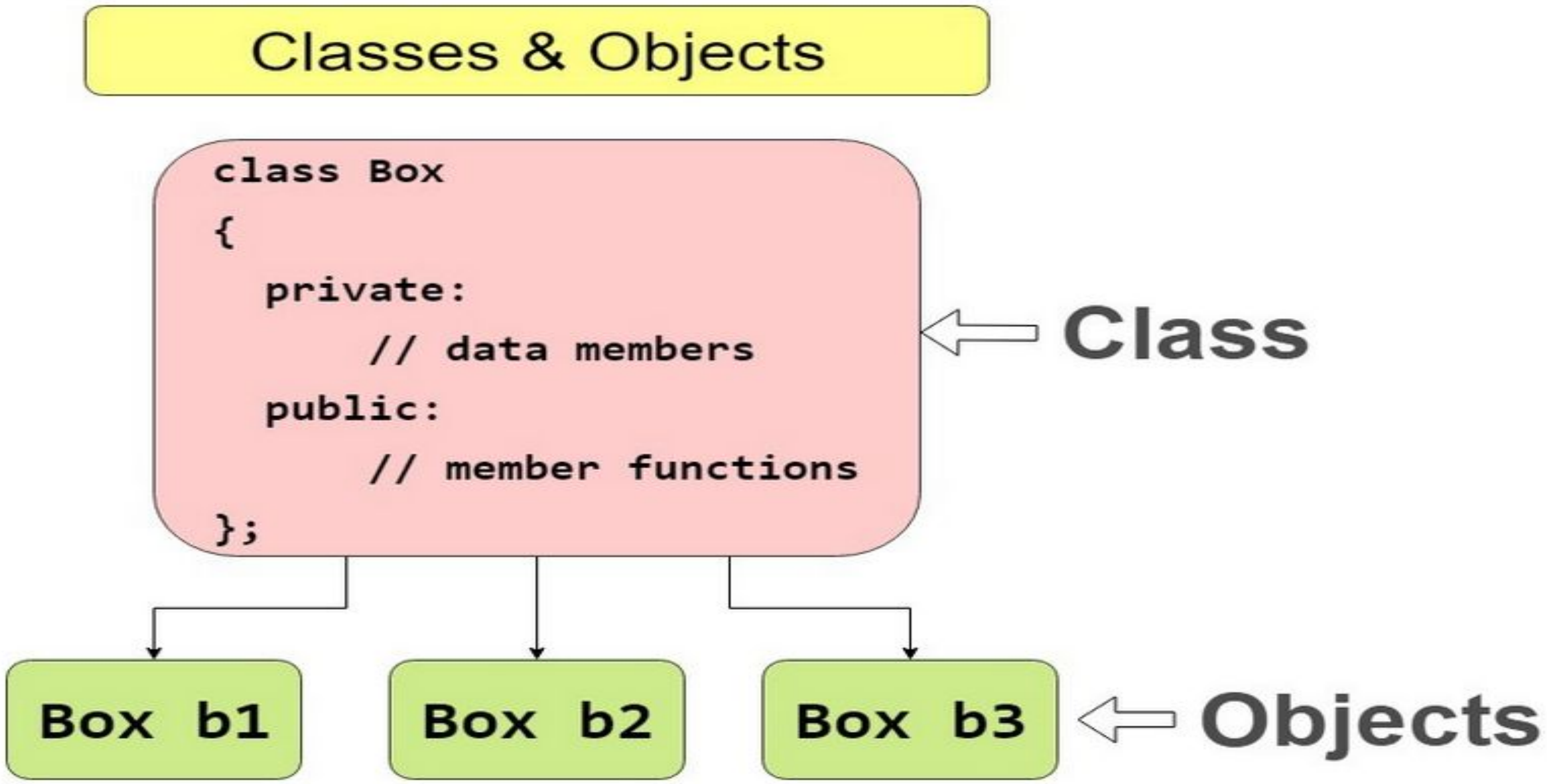
Classes and objects

- Last week we studied functions (more specifically user defined functions)
- Built in functions are functions that are found natively in the code to do a specific task.
- We essentially learned that we can make our own functions and it in be used in a modular way.
- Data types are built in to c++, they are found natively in the code.
- Similarly we can also write code such that we can create our own composite data type.

Classes and objects

- Classes are user defined data types.
- The entire point of a class is to help association of different objects and their features.
- In the long run it is rather hard to create variable names for an extremely large project and have no blueprint.
- Classes ensure that there is some level of hierarchy and control over the different variables.
- It also has some useful properties which makes it easier to form associations with similar objects.

Classes and objects ref https://simplesnippets.tech/classes-objects-in-cpp/



```
class Box
{
  private:
      // data members
  public:
      // member functions
};
```

Classes and objects

- Data members: Simply put these are the variables that are being used
- Member functions (or methods): These are the functions that are being used inside classes.
- Refer to classExample1.cpp and classExample2.cpp in Week 5 of our github page
- Access specifier lets you define the access for a variable.
- In c++ we use private and public to define access.
- public - the members are accessible anywhere.
- private - the members are only accessible in the class

Methods: Accessors vs Mutators

- The following examples we used are not indicative of ways we store values in these classes.
- Typically we have functions which assign values to data members.
- They are rarely declared within the class itself.
- We have functions which will change the values for us; These methods are Mutators
- If we have to only access the values i.e not change them but just use them; these functions are Accessors.

Exercises

1. Write a program to read information from the problem1.txt file in our github and find the average of the values.
2. Write a program with a custom class of your choice and give it some meaningful variable names and assign values.