Lab 2: Palindromes

Valentino Lei'a
CS_124
Fall 2020

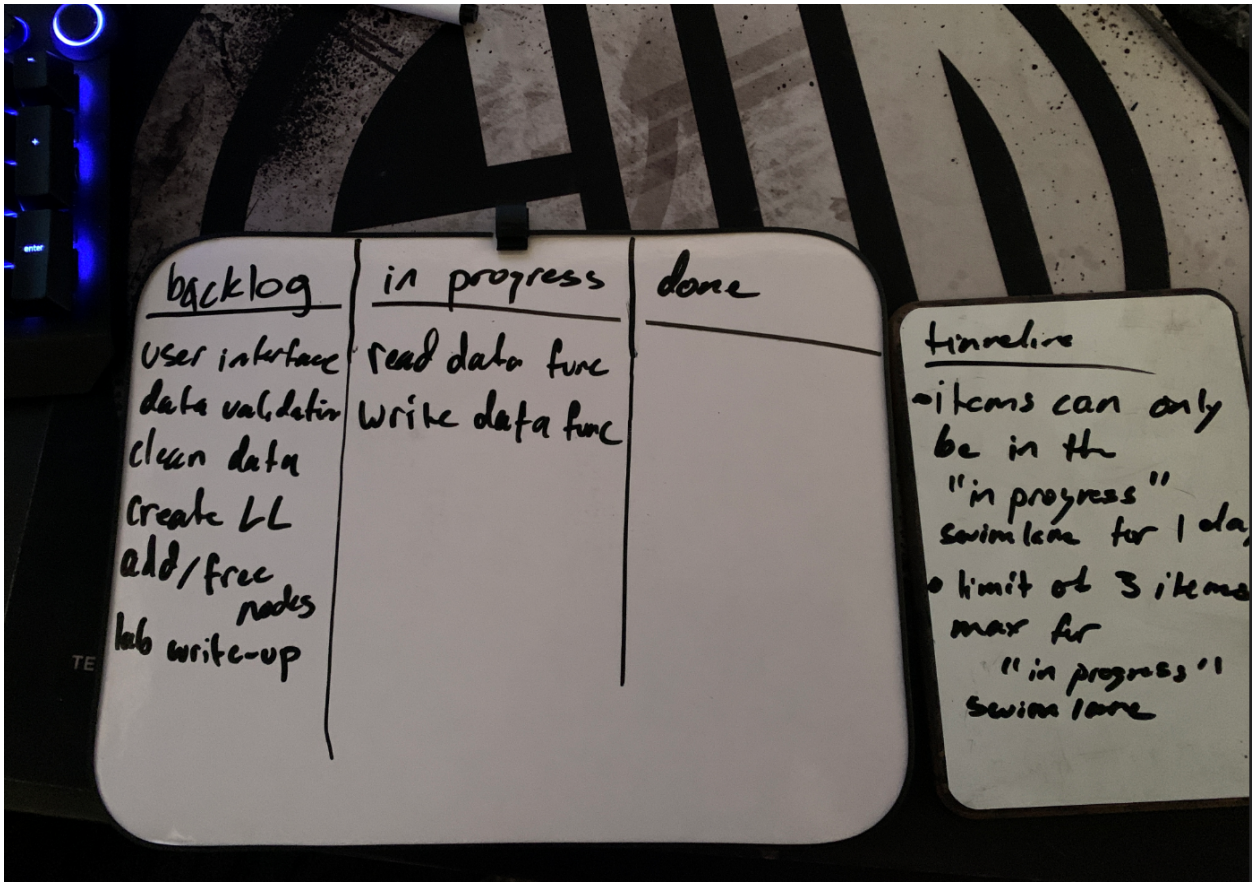
Purpose:

- Similar to our previous lab, the purpose of this exercise is to provide the students with an opportunity to implement the new concepts we've learned during this course up until now, along with our existing knowledge base of C++ concepts. Those being:
    - Stacks
    - Queues
    - Thinking about data structure (how to store the data, ie. arrays or linked lists etc.)
    - User interaction (cin, cout, data validation, user friendly messaging, etc)
    - Reading files into memory
    - Writing memory into files
    - OOP concepts (encapsulation specifically)
    - Fundamental C++ (function prototypes, variables, loops, etc.)
- Additionally we're provided another opportunity to present the finished product to a customer (professor) to simulate real world scenarios for when students get to begin their careers.

Plan & Organization:

- **What input does the program use? Why?**
    - The program is going to take a text file as input during the live demo because it's a requirement given by the professor
    - The end user's name is going to be taken as input from the console because it's a good user experience to ask the user their name and say hello for this program
    - The user will have the option to select between choosing a file or typing in their own palindrome, both of these options are input for the program and are requirements
    - A string provided by the user is going to be input that meets the requirement of this assignment
    - Strings from a .txt file is going to be input that is stored in memory as specified by the requirements
    - The response of the user wanting to repeat testing palindromes is input that is required for this program
    - BONUS: the user can upload their own text file
    - To touch a bit more on the why of all of these inputs, it's just a good user experience to be able to meet the need or common interactions that they will have with the program. You never want an end user to feel like they can't execute something with the product they're using and feel unsatisfied with the program.
- **What variables will you need? Why?**
    - Definitely going to need the string class to use strings when storing the ***strings*** from the file and user input
        - A little simpler than using cstring :)

    - I plan on using linked lists as a data structure to store palindromes in an ordered list
        - This will make it pretty easy to implement stack and queue for this assignment seeing as we got practice with it in a previous assignment :)
    - Plan on using some integers when I use for loops, or as a counter when I use a while loop, depending on the situation.
    - Definitely going to need an object of the fstream class to read and write files
    -
- **<u>What constants, if any, will you need? Why?</u>**
    - Constants, hard maybe for now depending on how I plan to clean data and store file names that already exist in the file path before runtime
- **<u>What functions do you use? How did you decide what operations go into each function?</u>**
    - A class that mainly handles user interaction
        - Functions for getting user input and printing to the console
        - I plan to have a function that validates user input with error handling
            - Overloaded to handle different data types
                - I've only used templates one time before so I still use this method (pun intended)
        - Setters and getters if needed
        - Constructors for each class
    - A class for data manipulation
        - A constructor that creates a linked list
        - Function that modifies that list (add in order, end, begin)
        - Function to prints the list
        - Deconstructor to free the allocated memory
        - I'm going to have another function that removes unwanted characters from a string
        - One function to read data from a file into memory
        - Similarly, a function to write data from memory into a file
- **<u>How do you decide that your program is complete?</u>**
    - I decide that the program is complete by first meeting every requirement listed in the requirements document.
    - If every criteria has been satisfied, then I can move onto nice to have functionality if time allows. Those things being the following:
        - Is there any undesired or odd functionality that exists
            - Does it need fixing or to be addressed?
        - Was there anything I "cheap'd" out on in terms of bad coding practice but just went ahead for time management
            - Can I go back and fix it to make it more efficient (cheaper or faster)


backlog
in progress
done
User interface
data validation
clean data
create LL
add/free nodes
lab write-up
read data func
write data func
timeline
- items can only be in the "in progress" swimlane for 1 day
- limit of 3 items max for "in progress" swimlane