Community College of Philadelphia

CSCI211 – Data Structures and Algorithms

Doubly Linked List

Overview

Using C++, Java, or Python, design a program that creates a **doubly** linked list, where each node contains an object with information about one of Philadelphia's fire stations.

Provided with the homework assignment is a CSV file containing entries for Philadelphia's fire stations. Each line/entry in this CSV file contains the data needed for the object's two fields.

Your program must read in the data contained in this CSV file to set the fields/attributes of the objects stored in your linked list's nodes. Using this data, your program will instantiate objects and append these objects into the nodes of the linked list.

See the following pages of these instructions for details about the necessary classes/objects and an outline of what will happen in your main program.

If you need a refresher on reading files and tokenizing Strings, here are some resources:

- C++
  - Reading/Writing CSV Files | Tokenizing Strings
- Java
  - Reading/Writing CSV Files
- Python
  - Reading/Writing CSV Files

You can use the above resources or other resources for parsing the CSV file, but be sure to leave a comment crediting the source if you do. *How* your program reads the data from the CSV file into your objects isn't being graded, just that the program *is* reading the data from the CSV file into your objects.

Object

FireStation
-stationNumber : int -location : String
+FireStation(stationNumber : int, location : String) +getStationNumber() : int +toString() : String

The class can be implemented as either a struct (C++) or a class and must contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram.
    - The **toString** method must return a string in the following format:

```
Station Number data - Location: data
```

Neither the program's output nor the string's text need to be in bold; The bold font used in the examples above are only to emphasize where the data should be in the string that your toString method returns.

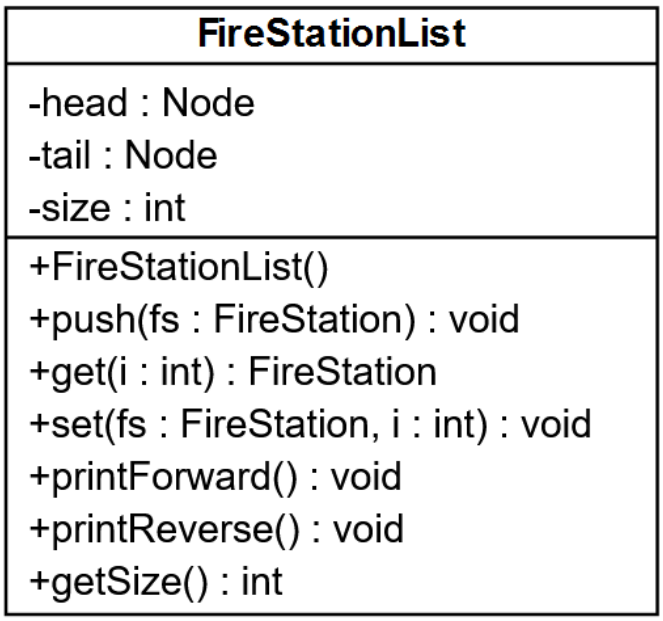
Node

Node
-fs : FireStation -next : Node -previous : Node
+Node() +setFireStation(fs : FireStation) : void +getFireStation() : FireStation +setNext(n : Node) : void +getNext() : Node +setPrevious(n : Node) : void +getPrevious() : Node

The class can be implemented as either a struct (C++) or a class and must contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram.
    - The constructor must initialize all fields to **null**.

Linked List



The class for your linked list must contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram:
  - The constructor must initialize the fields to **null**, and the length to 0.
  - The method that pushes a new object into the list will append it to new Node at the end of the list.
  - The method that retrieves an object from the list will do so using the provided position/index.
  - The method that accepts a new object will store it at the provided position/index in the list.
  - The method that traverses the list from head to tail will print each object's number and location using its toString method.
  - The method that traverses the list from tail to head will print each object's number and location using its toString method.
  - The method that returns the list's length/size.

Main Program

In a main function, complete the following:

1. Instantiate a new instance of your linked list.
2. Have the program read the CSV file's data:
    a. Parse the fields from the line/record.
    b. Instantiate an instance of a new object, providing its constructor with the data parsed from the current line/record.
    c. Append the newly created object to the end of the list.
    d. Repeat for all entries.
3. Sort the list by station number in ascending order.
    a. Implement a sorting algorithm of your choice.
4. Call the linked list's method to print the contents of the list.
5. Call the linked list's method to print the contents of the list in reverse.
    a. Calling these two functions will show that all Nodes are properly connected both ways.

You can do all the above in the main method or break it up into separate functions that the main method calls.

Other Items

- **Important: You may not use a List, ArrayList, Vector, or similar data structure built-in to your chosen programming language in place of your linked list class. You must implement a linked list and its necessary components/classes yourself.**

- Your classes/structs must closely match the UML class diagrams provided in these instructions. Slight deviations in field, method, parameter names are OK, but there is no need to include additional methods or fields in these structs/classes; Access modifiers (public/private) must be as they are seen in the UML class diagrams.
  See here if you need a refresher on UML class diagrams

- **Do not change or modify the CSV file**. I will be using the provided CSV file when I test your program.

- Be sure to use comments to adequately document your source code.

Grading

See Assignment Rubric in Canvas.