Wingtra coding work sample

This coding work sample can be worked on and compiled with Qt and opened in QtCreator (to see the UI) or you can just work on it using pure C++ to fill in the currently empty functions and add your solution code without seeing the UI. Both should work well and it's up to your personal preference and experience. Even if you decide to open it in QtCreator, feel free to not use the Qt framework in the code that you add and stick to pure C++ if you are more comfortable with that.
Hint if your environment isn't set up: For fast testing of pure C++ solutions, think about just opening the code in something like cpp.sh.

Feel free in general to use any resources at your disposal, such as google or reference books while working on this.

Note:

- We expect you to spend not more than 4 hours on this task, but you have 6h total until you should send us back your solution to leave some buffer for unforeseen problems.
- Completeness of each task is NOT the most important, we will focus on the discussion during the interview. We prefer you think about each task once instead of completing one of the tasks perfectly.
- If you get stuck or run out of time, feel free to skip parts, add only comments about how you would approach it or use pseudocode to describe your approach.
- We are also happy to have a discussion about more involved solutions / different software architecture approaches you would have chosen given more time or after a better understanding of the problem at the end. Feel free to add some suggestions in a comment or in a separate text file.
- The worsample is divided into an actual coding part (A) and a more high level system design (B) part. We expect the coding part to take about 2h and the design part to take about 2h, so leave enough time to work on both tasks.

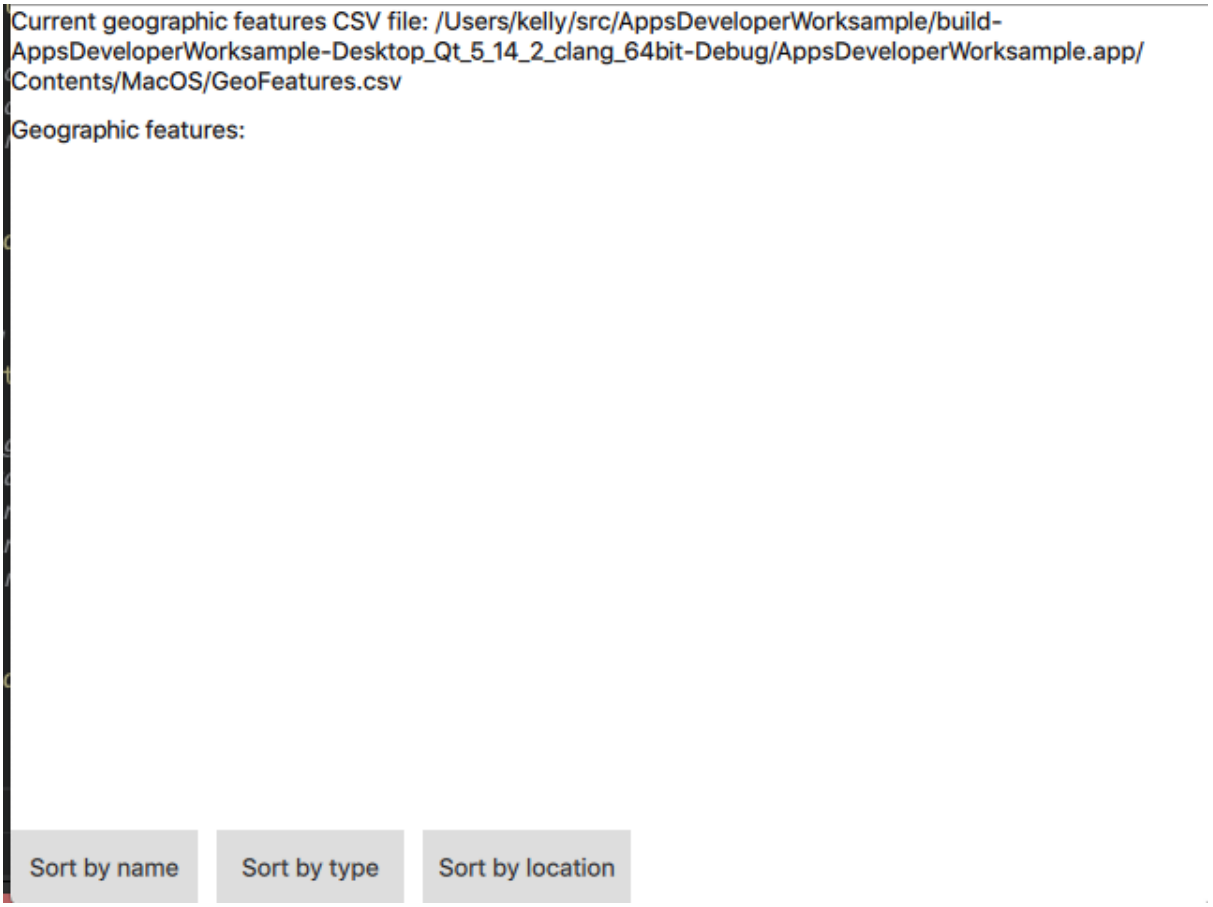
Deliverables:

- At the end of your 4h window (or whenever you feel done earlier) send us an email with a zip file attached containing again the code, now including your solution and any additional solution docs.

A. Geo-feature parsing (coding assignment)

Introduction

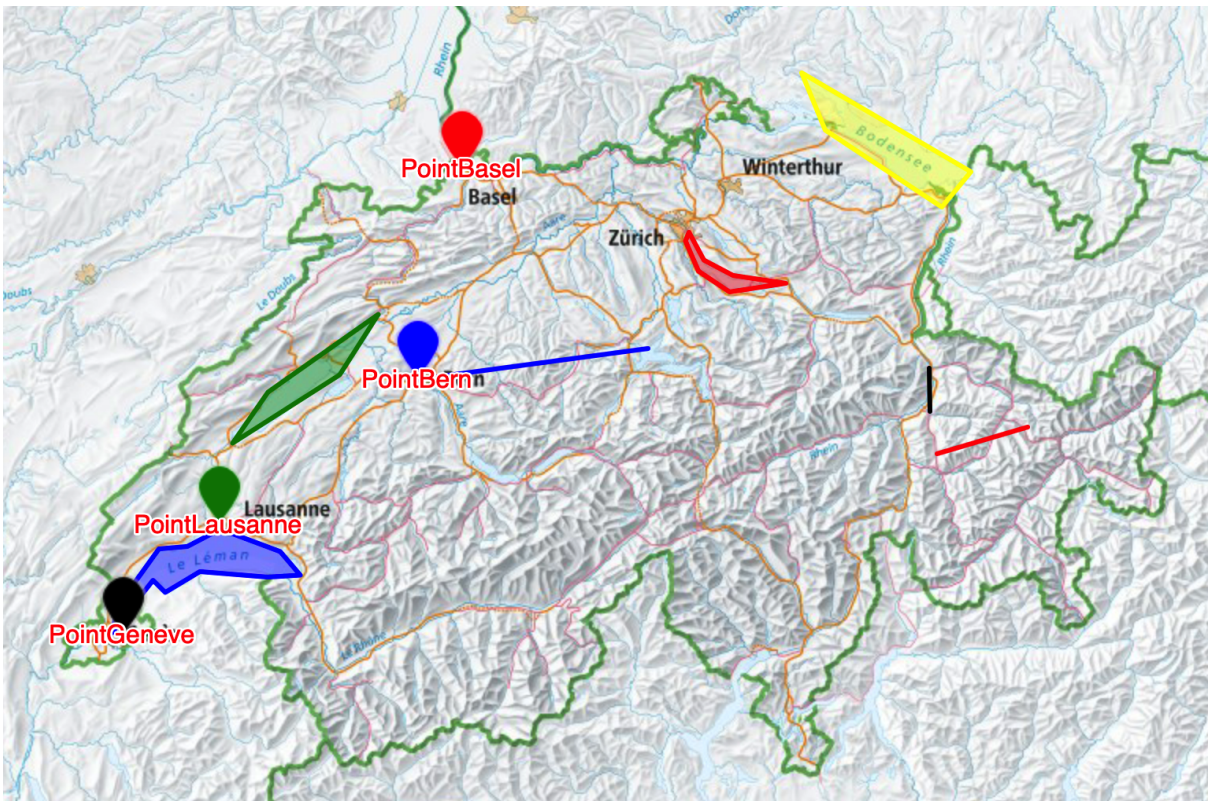
You are going to work on a small app that loads a CSV file, parses all geographic features contained in the file, and displays them in a list. This list can be sorted according to different criteria by pressing the buttons on the bottom of the UI. The empty UI looks like this:



The CSV file looks like this:

# name	type	color	coordinates
PointGeneve	point	black	6.19197585755413,46.21555604408079
PointLausanne	point	green	6.581561911548153,46.53517792203163
PointBern	point	blue	7.40250576313333,46.95444995969298
PointBasel	point	red	7.588749864926389,47.555419107015524
LineBernLuzern	line	blue	7.40250576313333,46.95444995969298 8.36956653056154,47.03951201666247
LineLandquartChur	line	black	9.541317231702296,46.967781445005514 9.539762365808768,46.844081001218186
LineLenzerheideDavos	line	red	9.94387398907487,46.79093716972668 9.564453481598418,46.72214184407179
PolygonLacLeman	polygon	blue	6.151012676444263,46.22139383926812 6.173813174887909,46.2999237654027 6.337921486991811,46.464478184711815 6.454944747441139,46.4727511971393
PolygonLacNeuchatel	polygon	green	6.638383033046999,46.769488468975844 6.78388198232887,46.91659643768211 7.2425356062623285,47.14093330674841 7.082205368854951,46.967391122710
PolygonZurichsee	polygon	red	8.546827397478946,47.37122331403928 8.608614162590403,47.290555143608394 8.730474363428 02,47.24517517190927 8.950378903123033,47.220008601737
PolygonBodensee	polygon	yellow	9.034367587349077,47.82158296219738 9.739435001280201,47.52637436114077 9.612517890621518,47.4276062449266 9.129237641284982,47.656082923676

Each line is a geographic feature, which is the shape of either a polygon, a point or a line. Apart from this shape type, each geographic feature also has a name, a color and of course coordinates describing its shape. This is what it would look like if all the shapes in the file were displayed on a map:

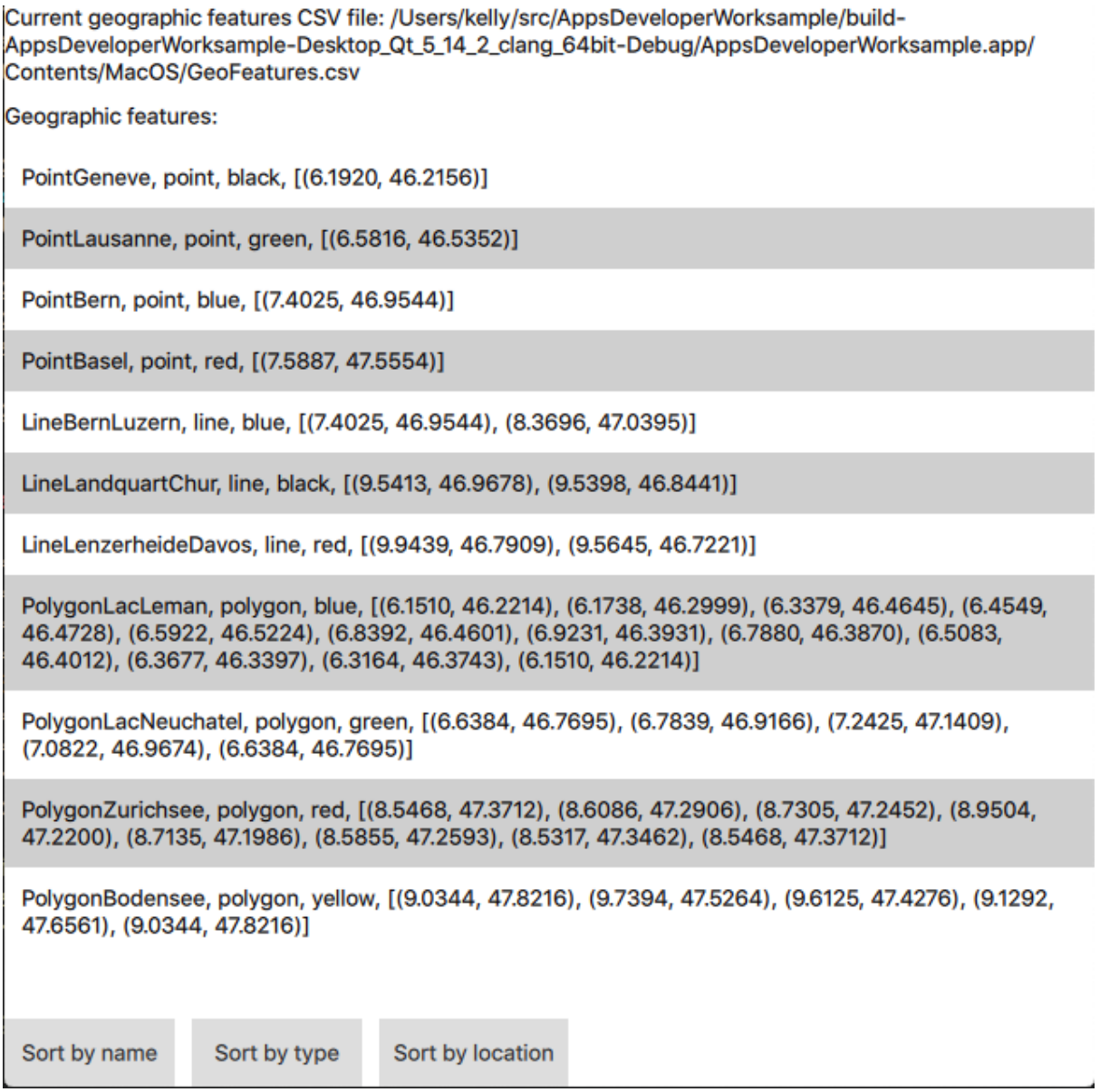


Start by:

- opening AppsDeveloperWorksample.pro in Qt creator or opening the project in your favorite code editor
- (Compile the app if you want to see the UI)
- Open the GeoFeatureParser.cpp file to get started

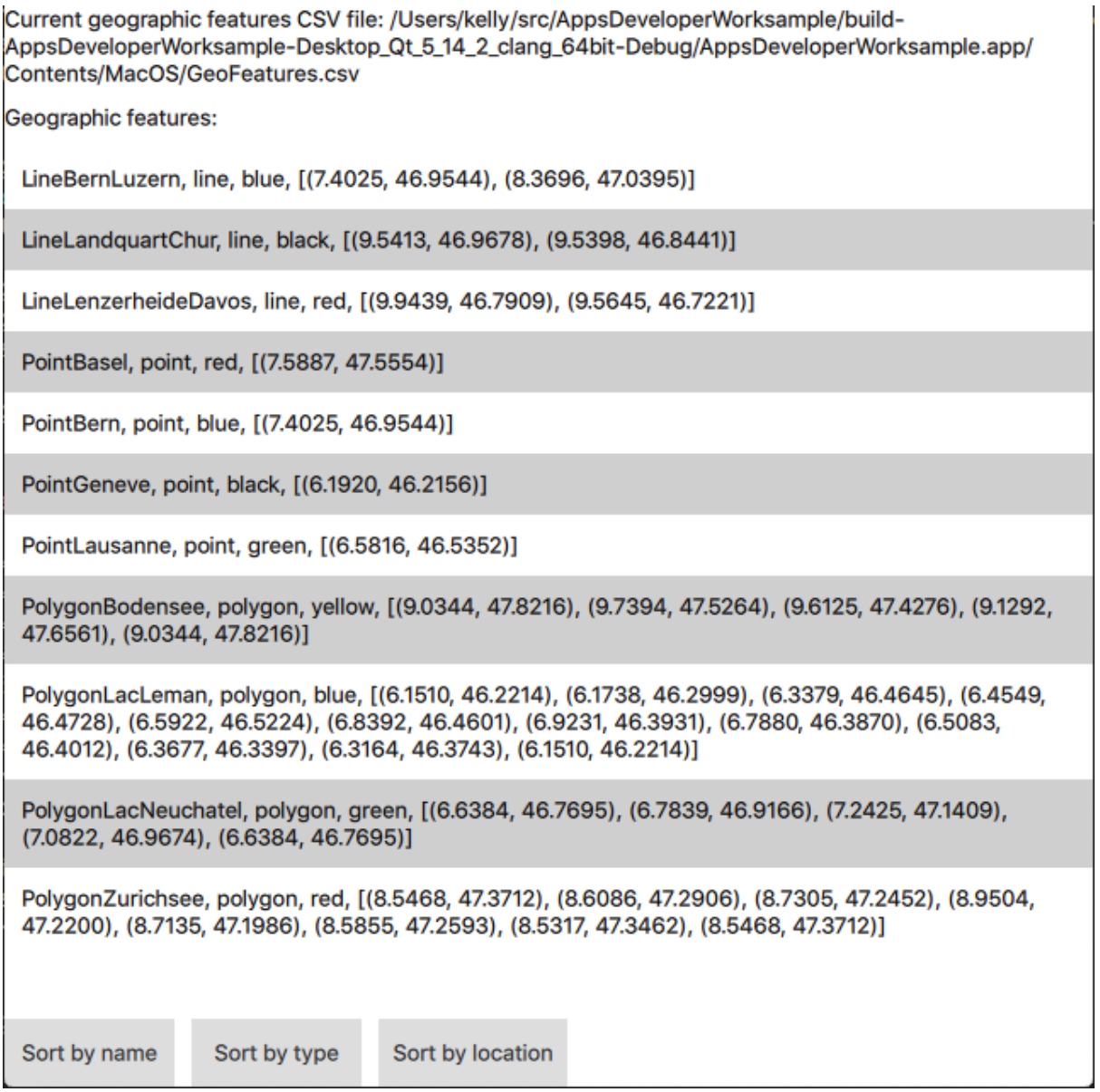
Task 1

- In the file GeoFeatureParser.cpp look at the function _parseGeoFeatureCSVFile
- Read the TODO here and implement the required functionality
    - You can of course add more functions, classes, files, … as desired or needed
- Desired output:
    - Right when you start the app it should now look like this:



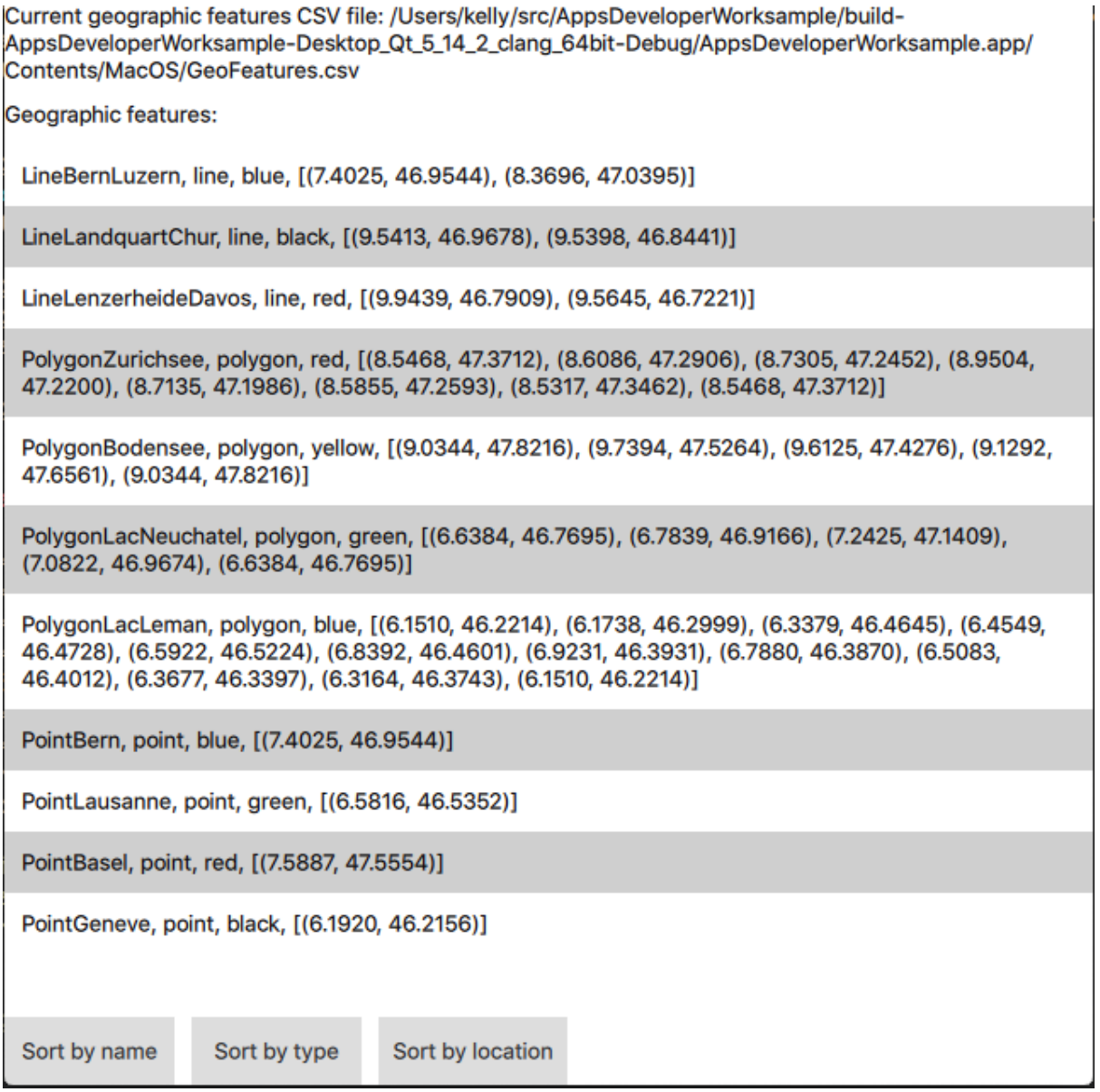
Task 2

- In the file GeoFeatureParser.cpp look at the function sortByName
    - This function is called when clicking on the first button in the UI
- Read the TODO here and implement the required functionality
    - You can of course add more functions, classes, files, … as desired or needed
- Desired output:
    - After clicking the first button “Sort by name”, the app should now look like this:



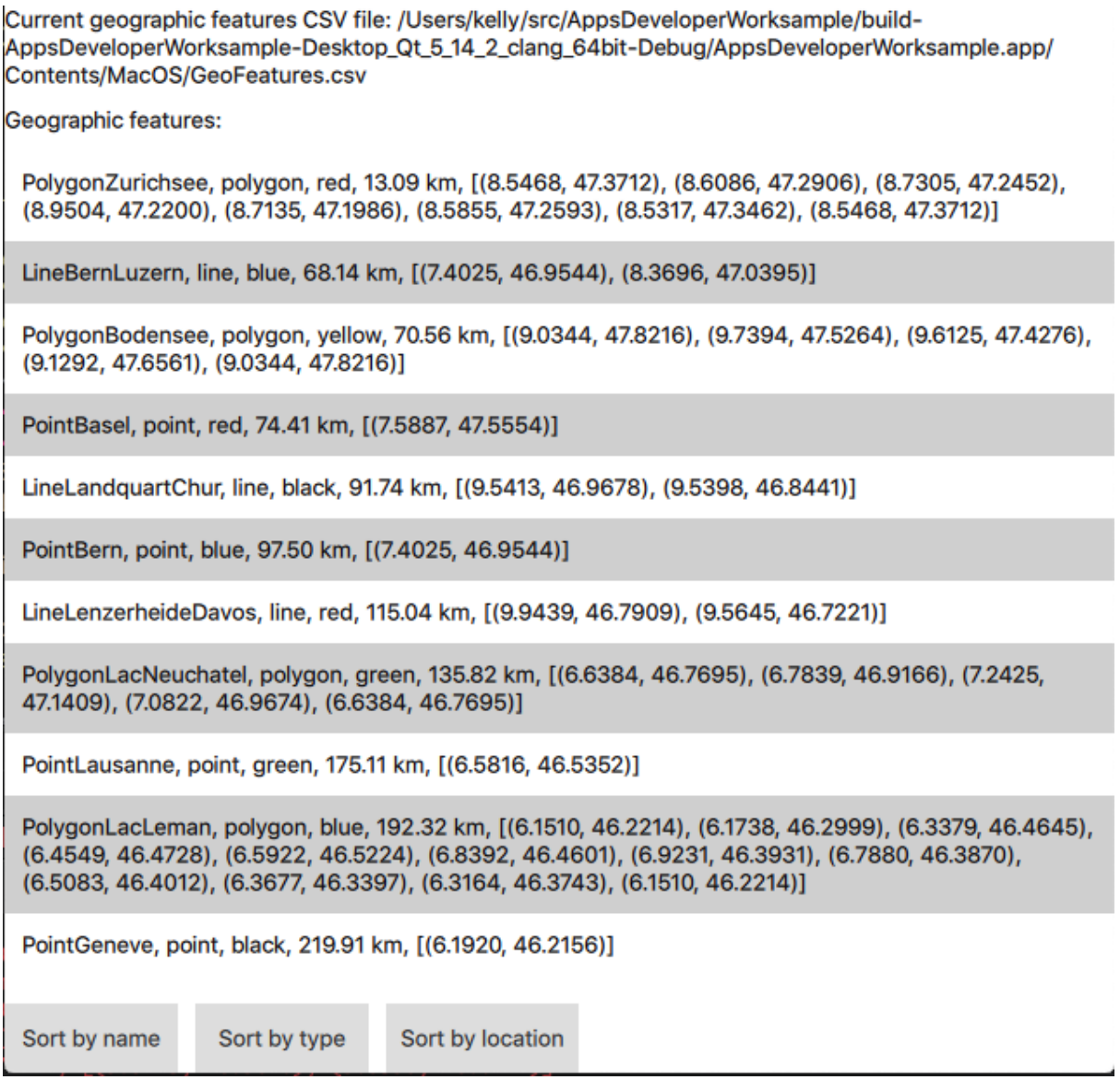
Task 3

- In the file GeoFeatureParser.cpp look at the function sortByType
    - This function is called when clicking on the second button in the UI
- Read the TODO here and implement the required functionality
    - You can of course add more functions, classes, files, … as desired or needed
- Desired output:
    - After clicking the second button “Sort by type”, the app should now look like this:



Task 4

- In the file GeoFeatureParser.cpp look at the function sortByLocation
    - This function is called when clicking on the third button in the UI
- Read the TODO here and implement the required functionality
    - You can of course add more functions, classes, files, … as desired or needed
- Desired output:
    - After clicking the third button “Sort by location”, the app should now look like this:



B. App metadata collection framework (design assignment)

- Imagine now a much more mature app than our sample app here. The user can for example load multiple such files, open them and display the geographic features on a map. Individual geographic features can be moved around on the map and interacted with. The user can design paths connecting these features and get statistics about the complexity and length of these paths, etc.
- To develop further features, we want to understand better how the users interact with the app. For that purpose, we would like to add our own cross-platform module for saving and uploading metadata events to our own server. Metadata events should be easily triggered in the app when the user does a specific action and can have multiple attached parameters describing the event further.

- There can be times when many such events would have to be saved in a short time (f.ex. a burst of approx. 500 events at once) while the app is running, and other times where we would just want to save an event approx once every 5 seconds.
- The app can be used both online and offline and a strict requirement is that we never lose any metadata, even while offline, given that users will be online at regular intervals.
- Before implementing such a framework with your team, you are asked to come up with a detailed suggestion of the system architecture.
- Please submit a separate document containing:
    - Your understanding of the problem and a list of resulting functional and non-functional requirements.
    - A high level description of your suggested solution.
    - An architecture diagram for such a module.
    - For each component describe the basic functionality using example pseudo code.
        - If time allows, feel free to add actual code snippets also to the sample app.
    - An explanation for why you chose this particular solution approach and what the tradeoffs are to other solutions / what the difficulties are you would most likely face with this solution.