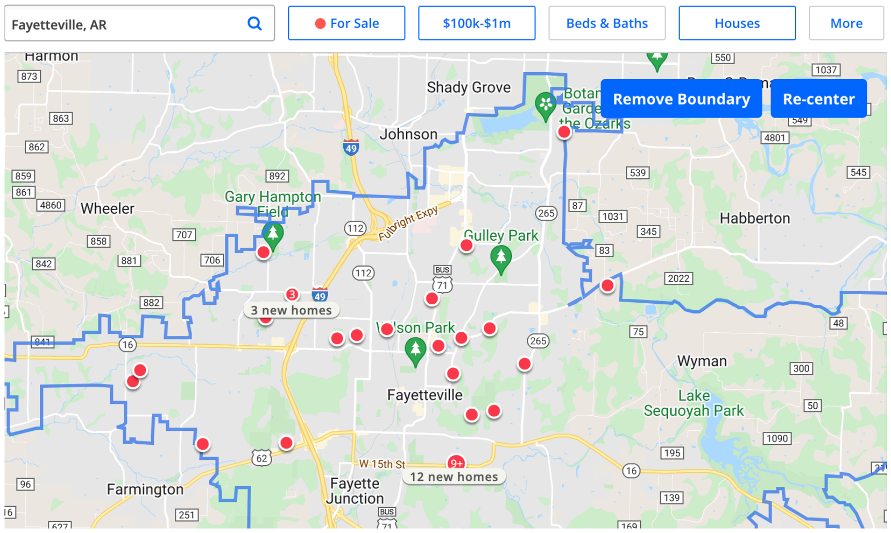
CSCE 2014 – Programming Project 5

Midpoint Due Date – 10/31/2022 at 11:59pm
Final Due Date – 11/07/2022 at 11:59pm

1. Problem Statement:

The purpose of this project is to give students experience with sorting algorithms. In particular, you will be required to implement two different sorting algorithms. The first sorting algorithm can be either "insertion sort" or "selection sort". The second sorting algorithm must be either "quick sort" or "merge sort". Instead of sorting an array of integers or floats, you will be sorting a vector of RealEstate objects based on their attribute values. Detailed requirements for the project are included below.



Screenshot from Zillow.com showing houses in $100K to $1M range.

In this project, you will be given a fully implemented RealEstate class that stores 10 pieces of information about recent real estate listings (house_number, street_name, city, state, zip_code, asking_price, number_bedrooms, number_bathrooms, house_size, lot_size). This class has the normal constructor functions and getters and setters. It also has four methods for reading and writing RealEstate information from txt files or csv files. You will also be given a small main program that opens and reads an input txt file to create a vector of RealEstate objects, and then loops over the RealEstate vector to create an output csv file containing the same real estate information. Finally, you will also be given the following data files (houses_big.txt, houses_big.csv, houses_small.txt, houses.small.txt).

Task 1: Create a RealEstateDB class

Your first task will be to create a RealEstateDB class that contains a vector of RealEstate objects. Your class should contain the following input/output methods:

ReadTXT – Read a txt file into the RealEstate vector.
ReadCSV – Read a csv file into the RealEstate vector.
WriteTXT – Write information in the RealEstate vector to a txt file.
WriteCSV – Write information in the RealEstate vector to a csv file.
Print – Print all RealEstate information to the screen.

Task 2: Implement two sorting algorithms

Once you have the input/output methods in your RealEstateDB class working you can implement your two sorting methods. As noted above, you must implement two different algorithms. The first must be "insertion sort" or "selection sort". The second must be either "quick sort" or "merge sort".

You get to decide which of the data fields within the RealEstate class to compare when implementing these two sorts. Since our real estate data is all from one city and state, you should choose two fields from the remaining eight to compare. For example, you could implement a SortPrice method using "selection sort" and a SortHouseSize using "merge sort".

Task 3: Implement a main program

Your final task will be to implement a main program that demonstrates the correctness of your four input/output methods and your two sort methods. You can do this using a menu or with a sequence of user prompts. Your goal should be to create output files sorted according to your chosen attribute fields.

2. Design:

TBA

3. Implementation:

To implement your project, you should break your code down into **five** files: real_estate.h, real_estate.cpp, real_estate_db.h, real_estate_db.cpp, and main.cpp. For this project, you are welcome to borrow and adapt any of the sorting examples on the class website.

As always, it would be a good idea to start with "skeleton methods" to get something to compile, and then add the desired code to each method incrementally writing comments, adding code, compiling, debugging, a little bit at a time. Once you have the methods implemented, you can create a main program with s simple menu interface that calls these methods to complete your project.

Remember to use good programming style when creating your program (good names for variables and constants, proper indenting for loops and conditionals, clear comments). Be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

For this project students are required to compile and test their program on Linux. Once you have logged into turing.csce.uark.edu, copy your source code into a folder called "project5". Go into this folder and type in the command "script". This will create a "typescript" file that records all user input and program output until you type "exit".

Compile your program using "g++ -Wall *.cpp -o main.exe". Then run your program by typing "./main.exe" to verify that your program operates correctly for all of the requirements listed above.

When you are finished testing your program, type "exit". Copy the "typescript file back to your personal computer to be included with your code and project report when you submit your completed project into Blackboard.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report to be submitted electronically.

6. Midpoint Submission:

For the midpoint submission, you should upload your code after you are approximately 1/3 to 1/2 way through this implementation process. It is OK to have skeleton methods that simply print debugging messages at this stage. The main thing we want to see is code that compiles and is part of your final solution.

7. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above, copy all of your source code, your typescript file, and your project report into a folder called "project5". Compress this directory into a single ZIP file called “project5.zip”, and upload this ZIP file into Blackboard. The GTAs will download and unzip your ZIP file and compile your code using “g++ -Wall *.cpp” and run your program to verify correctness.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

8. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are **not** allowed to distribute code to each other, or copy code from another individual or website. Students **are** allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance. This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a zero on the programming project, an XF in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.