

Register Container Persistence

Purpose:

This exercise is intended to give you experience with storing and retrieving information in data stores. We will use one particular technique for our register—creation of a text-editor accessible file with a codified format. We will discuss other possibilities.

This exercise is intended to be a simple and straightforward modification of the previous exercise. You may wish to use the previous sample solution as a starting port.

What to hand in:

Hand in a listing of the program (which must be understandable and formatted in a reasonable style), and a sample run of the program. Please ensure that your test program *thoroughly* tests the required classes.

The program will include a Register header file, one or more Register implementation files, and a test program that will store different types of entries in the register and ensure the functions work on them, demonstrating appropriate virtual behavior

You should create an `ioregister.h` and `ioregister.cpp`, and add them to the previous project, ensuring that your program can create and write out the various elements. Your `testRegister.cpp` (or otherwise named) main program calls the functions defined there, and populates and saves the register.

Please capture your homework in a single file for submission, in an easily reviewable order.

Problems:

Register Persistence

Create functions in `ioregister` that allow

- (a) an entry from a file used for persistence to be parsed; and
- (b) a persistent file to be written from the contents of a Register.

Note that the “write” functions (as opposed to the “print” functions) in the sample solution address the mechanics of preparing the output.

The function of (a) above is a *factory* method for creating entries.

The two functions are independent

Notes:

For testing purposes, you may wish to have the name of the read and written registers be different so they can be compared as well as repeating the program run.

Also note that the functions in `ioregister` might not be in a class.

This exercise builds from the previous one in a straightforward way. The sample solution to the previous exercise provides most of the functionality, with the test program having a few changes.

Writing to the register is the easiest of the two functions, since the classes are already prepared with a textual output preparation function (`Write`).

Extracting these values from the file and matching them up to constructor arguments is a little more difficult—but functions to parse a date and money are provided in the sample solution to get you started on this.

Evaluation criteria:

ioregister interface (.h file)	35%
ioregister implementation (.cpp file)	35%
Test program (.cpp file) and output	30%