C++ for Finance - Assignment 2 - 24th March 2020 (due: 12 noon 27th April 2020)

Laurence A. Hurst - University of Birmingham - Academic year 2019-20

Your assignment

For your assignment you need to complete the following coding project, and then write a short report on it. Both the code and your report will be marked and both must be submitted before the deadline.

Coding task

Your task is to write, in C++ utilising any language features available in C++14 (as taught in this module) and its standard library, a personal financial planning program. You may not use any libraries that are not in the C++14 standard library in your solution (everything shown to you in the module is part of the C++ standard library).

A group of clients are planning for their retirement and want your program to manage an amount of capital for them. The clients are only interested in very low risk products, so you will be only using (fictional) interest-bearing products that are similar to government bonds, with their capital. Your program is hypothetically going to run continuously until the last client retires, taking a continuous feed of product data to manage the initial capital and maximise return for them. In reality, this feed is provided in full and the program will run through the many years of data and transactions in a few seconds (or less) so this assignment can be marked.

Your program will be provided with two files. From these files your program must read in a list of clients, a list of financial products and produce output files, one per client, which is their personal financial plan.

Your program must move through the data from the brokerage firm sequentially - it may **not** look ahead to see what products are coming to decide what it will do next (we are imagining the data feed is a continuous stream that arrives throughout the years between 5th April 2020 and each persons retirement date).

The rules are as follows:

- Your program cannot make decisions based on what comes next in the data feed. Your program will have to decide whether to invest all (or the maximum), some or none of the clients available money in each product.
- At no point may any client go into negative available capital (but they may invest all of it - so have 0).
- No re-sale of products are permitted during the tie-in period, from the date of the investment. If the product's maturity date is reached during the tie-in it may be redeemed on that date for capital plus interest for each completed anniversiary.
- Products may be resold, after any tie-in period, for the original capital plus accrued interest. However, if resold before a monthly or annual anniversary (depending on if interest is paid monthly or annually) result in zero interest being paid for that period. (e.g. monthly product invested in on the 1st January and resold on 25th March will get 2 months interest paid, hypothetically interest from 1st March would be paid in full to the new owner on the next anniversiary - i.e. 1st April.)
- If the monthly anniversary falls after the end of the month (e.g. 29th-31st in February, or 31st on months with 30 days) interest is paid on the last day of the month.
- The first input file will be called 'clients.csv' or 'clients.tsv'. Your program must detect and display an appropriate message if both are found.
- The second input file will be called 'feed.dat'.
- Both files will be in the same directory as your program.
- You programs must detect malformed or missing input files and display an appropriate message.
- All clients intend to purchase an annuity on the day of their retirement, so all money must be available as capital at that point. Any money still invested (even if not tied-in) will be seen as not part of the clients capital and excluded from the bonus mark competition.

The first file is an export from a client management system and will be either tab or comma-separated (your program must be able to process either format - the file-name may be used to infer which is being supplied) and will consist of the

following fields, in this order (the equivalent C++ data type is specified afterwards):

1. Name (string - may contain spaces)
2. Current capital in pence (int) (i.e. cash immediately available)
3. Expected retirement date (string in the form YYYYMMDD - e.g. 20201225 is 25th December 2020)

The second file is a chronologically ordered (by product start date) data feed from a brokerage firm, it has fixed width fields (the data consists of a precise and fixed number of characters within each file). It will consist of the following data, in this order:

1. Institution code (4 alphabetical characters)
2. Product code (8 alphabetical characters)
3. Product AER interest rate * 100 (5 digits - for example 00522 is 5.22%)
4. Annual or Monthly interest (1 character - A or M respectively)
5. Product tie-in period in years (2 digits)
6. Product minimum investment in pounds (6 digits)
7. Product maximum investment in pounds (6 digits)
8. Product available from (8 digits - YYYYMMDD)
9. Product matruity date (8 digits - YYYYMMDD)

The output format for each client must be a CSV file that contains the following fields in this order for each investment/re-sale/redemption they should make:

1. Date of transaction (8 digits in the form YYYYMMDD)
2. Product institution (string)
3. Product code (string)
4. The word "Invest", "Sell" or "Redeem" (string)
5. The amount to invest or acquired via sale/redemption from that product in pence (int)

For the purposes of the assignment (to simplify the task for you):

- Years are always 365 days (i.e. ignore leap years).
- The start date for the exercise is 5th April 2020 - you should assume that is the date of the date extract from the client system.
- All interest is compound interest and is paid at the maturity of the product or on resale.
- Monthly interest is calculated per-calendar month, not daily, so the length of the month is irrelevant to the calculation.
- We are ignoring the possibility of any fees on products etc.
- all products are fixed term - redemption must be recorded in your output file at maturity or it is irrecoverably lost. For example, you invest £5,000 (500,000p) for a client on 2nd July 2020, the maturity is 30th June 2025 and the tie-in period is 2 years. The product may be resold at any point after 2nd July 2022 (2 years) or must be redeemed (with any interest) on 30th June 2025 or the capital and interest will be lost.
- This is finance, all fractional values are rounded **down** to the nearest pence at the time of the transaction.

You have been supplied with 2 correct sample input files to test with and verify your reading code. These are **not** the files that will be used to mark your code - do not make assumptions based on them.

To convert from AER to monthly gross rate for compound interest you may use this formula: $\sqrt[12]{1 + AER} - 1$ (so 5% AER = $\sqrt[12]{1.05} - 1 \approx 1.00407412378 = 0.407412378\%$).

This task can be completed using only what you have been taught on this course.

Report task

Alongside your code you must submit a short report about your solution. At a minimum it must address each of the following:

1. Briefly explain your approach to the problem and why you have chosen to implement that as your solution over any others you may have thought of or researched.
2. A summary of the algorithm as implemented in your code (you might find it helpful to include a mathematical description and/or a flow diagram).
3. An explanation of how efficient you think the **algorithm** you have used is and any efficiency considerations.

4. A reflection on the efficiency of your implementation (specifically your code/implementation, not the efficiency of the algorithm you have used) and any considerations, including any investigations/tests you may have conducted to asses this.

Please make sure to address the last two points individually.

Assessment

For this assignment your programs functionality will be automatically marked - make sure your program conforms to the rules above and produces the correct output.

This project is worth 40% of your overall mark for this module. Of that 40%, 70% of the marks (28% of your overall mark for the module) will come from your code and 30% (12% of your overall mark for the module) from your report.

Up-to 5 additional marks (maximum score for the assignment remains 100) will be awarded to the student or students whose solutions give the largest average return across all the clients in the marking data.

What you must hand in, by upload to Canvas

You must upload two files to Canvas by the project deadline:

1. A zip file containing all of the code for your project - refer to the Exercise 1 from week 5's exercise sheet for what you need to include with your submission (n.b. do not share your project code with your classmates).
2. A Microsoft Word (doc or docx) or PDF file containing your report.

The marks for your code will be split between functionality (it solving the problem set and the efficiency of the algorithm used) and the quality of the code. Code quality will be measured against what you have been taught on this course; judging style, understandability, efficiency of your implementation, maintainability and layout (this is not an exhaustive list of critera, nor is it an indication of the importance of any one aspect, but to give you an idea of what is being looked for).

Additional Important Information

Your report, code and any comments in your code must be written in English.

The code you upload must run unmodified in the CLion IDE as installed in the teaching lab computers in G04 - this is the environment we have been using throughout this course.

Your report file must be print-ready, it is your responsibility to ensure the report can be printed correctly without modification.

Your report must have a title page giving the assessment title and your university number.

In your report you may refer to your code by filename and line number(s), for example "line 30 in main.cpp" or "lines 30-35 in main.cpp", or include small code snippets if you wish. Make sure the line numbers are correct before you submit, especially if you have changed any code since writing your report.

Your report body must be no longer than three A4 pages (including any and all diagrams, tables and code snippets but not including the title page), using either Calibri or Times New Roman font and a minimum font size of 11pt.

The name of the report, and the zip file must be your university number. Rename the internal C++ project name to be your ID number. Your university number must also appear on the title page of the report and in a comment at the top of each .hpp and .cpp file. (marking is anonymous so your name must not appear anywhere).

The code in the zip file will be compiled and run to confirm it functions and correctly solves sample Sudoku puzzles.

To reiterate: **your student ID must appear in every source code file**.

Group work is not permitted

You are encouraged to discuss the assignment and ideas for approaches with other students but all code you submit must be yours alone. Clear similarities in code or in the report will be taken as evidence of group work (for instance if identical or very similar results for your assessment of your code are given, or if code is clearly shared).

Code closely modelled on that from other sources is not permitted

Code to perform this exercise, in C++ or other languages, may be found in various places and might, for instance, be downloadable from the internet. Use of any code taken, or cosmetically altered, from such sources (or elsewhere) is not permitted (with or without attribution). You must devise your own code from scratch.

Specifically excluded from this prohibition is code and examples in the lecture materials (including the exercise sheets and solutions), which you may refer to and model parts of your solution on without attribution.

You may, and are encouraged to, research general approaches to this task - but you must not use, or model your work on, any existing code.