Assignment 1 specification

Version 1.3

1 Outcomes

After completing this assignment, students will be able to:

- Implement the list structure
- Perform basic operations on list

2 Objective

Students are required to build a program on C ++ that reads information about the foreign exchange market [1], implement simple requests on this information and write the result in a text file.
To accomplish this task, students need to download assignment1.zip file, which contains the files:

- src/main.h, src/main.cpp: Defining main functions, such as: read a text file containing information about the foreign exchange market, invoke function and write the results to the result file. Students are allowed to read these files but are not allowed to change any content of these files.
- src/processData.h, src/processData.cpp: Implementing information processing on currency transaction. Students perform tasks through this file. However, students are NOT allowed to add any #include other than the existing #include in these files.
- test/input/1.txt, test/input/2.txt: These file contain information about sample currency transactions, program inputs. Each file contains multiple lines, one for each currency transaction information processing command. The processing requirements for each order are detailed in Section 3.
- test/output/1.txt, test/output/2.txt: These file contain the result of processing currency transaction information corresponding to input file.

After extracting, students have to translate the provided C++ code, then execute by typing the following command on the command prompt (Command Prompt / Terminal):
main test/input/1.txt test/output/0.txt

After executing the above command, students check again by comparing the contents of the file **test/output/0.txt** and **test/output/1.txt** using the following command:

- Windows: *FC test/output/0.txt test/output/1.txt*
- Linux/MacOS:*diff test/output/0.txt test/output/1.txt*

The result of the comparison must show that these two files are not different.
Students do assignment by editing the processData.h and processData.cpp files to organize data storage and information processing as required, test the program by creating new input on the input directory and the corresponding output file on the output directory.

3 Data processing command

3.1 General instruction

Each data processing order is a line on the currency transaction information file. Each command begins with a keyword (the word in bold in the description) and is followed by parameters (between < and > marks in the description). There is only one space between the command and the parameters. There are no spaces before the command keyword and no spaces after the final parameter. Some final parameters, which are placed in [and] in the description, are optional (appear or not appear in the command). When a command does not provide number of parameters exactly or does not have the same type of parameter as described or contains more extra spaces, the command will not be processed and the result will return -1. Otherwise, the command will be processed and return an integer value ≥ 0 as described in Section 3.2
The meaning of the acronym and the data type of the parameters is described as follows:

- TIME: an integer number (representing the time according to ISO standards) show the opening time of the session.
- OP: (opening price) a real number show opening prices.
- HP: (high price) a real number represent the highest price.
- LP: (low price)a real number represent the lowest price.
- CP: (closing price) a real number representing closing prices.
- BC: (base currency) a code (string) representing the purchase currency code.
- QC: (quote currency) a code (string) representing the sale currency code.
- CODE: a string starting with the letter C followed by a number representing the specific requirements of the command.
- INST: a string representing the command name keyword.

There are two optinal time's parameters in the command <TIME_A> and <TIME_B> (performing [<TIME_A> [<TIME_B>]]), the program will execute the command with other parameters and

- <TIME_A> ≤ <TIME> ≤ <TIME_B> if <TIME_A> and <TIME_B> are included in the command.
- <TIME> = <TIME_A> if there is only <TIME_A> in the command.
- Every <TIME> if there is no <TIME_A> and <TIME_B>.

Each order needs to be handled with a time complexity that does not exceed the rule in the "Complexity" column when N ≥ 100, where N is the average number of candles per currency exchange pair.

After finishing to process all instructions in the input file and write the results to the output file, your program must ensure to deallocate all dynamically allocated data objects and not to leave any memory garbage before ending.

3.2 List of commands

Requirement	Complexity	Description
INS <BC> <QC> <TIME> <OP> <HP> <LP> <CP>	O(N)	Add candle data to the data structure and return the number of candles after adding.Data of candle can be added by the time randomly, and if time overlaps, subsequent candle data can not be added to the data structure.
DEL <BC> <QC> [<TIME_A> [<TIME_B>]]	O(N)	Delete candle data which <BC> and <QC> values correspond to the <BC>, <QC> parameters of the command according to the time provided by <TIME_A> and <TIME_B>. The command returns the deleted candle data.
UPD <BC> <QC> <TIME> <OP> <HP> <LP> <CP>	O(N)	Change the <OP>, <HP>, <LP>, <CP> values of the candle with <BC>, <QC>, <TIME> values corresponding to the parameter. If there is no candle data matching parameters <BC>, <QC>, <TIME>, the program will return 0, otherwise, the program will change the values <OP>, <HP>, <LP>, <CP> of the corresponding candle in the data structure and return the number of changed candles.

ST <BC> <QC> [<TIME_A> [<TIME_B>]]	O(N)	Counts and returns the number of candles for the currency exchange pair <BC> <QC> with a spinning top [1] from TIME_A to TIME_B.
MB <BC> <QC> [<TIME_A> [<TIME_B>]]	O(N)	Counts and returns the number of Marubozu candles [1] of the currency trading pair <BC> <QC> from TIME_A to TIME_B.
DJ <BC> <QC> [<CODE>] [<TIME_A> [<TIME_B>]]	O(N)	Counts and returns the number of Doji candles [1] of currency exchange pair <BC> <QC> in the period TIME_A to TIME_B. If the command have a parameter <CODE> (a string starting with letter C), the program will count the Doji candle type according to the value of <CODE> as follows: • CODE = C1, count Long-Legged Doji candle. • CODE = C2, count Dragonfly Doji candle. • CODE = C3, count Gravestone Doji candle. • CODE = C4, counting Four-price Doji candle. Without the parameter <CODE>, the program will count all types of Doji candles.
EG <BC> <QC> [<CODE>] [<TIME_A> [<TIME_B>]]	O(N)	Counts and returns the number of Englufing candlestick group [1] of currency exchange pair <BC> <QC> in the period TIME_A to TIME_B. The <CODE> parameter is processed like the command **DJ** with the <CODE> code for the Engulfing candle group as follows: • CODE = C1, count bullish Engulfing candlestick groups. • CODE = C2, count bearish Engulfing candlestick groups.

Note: In this assignment, we assume that the length of the body of the candle or the shadow of a candle is considered small, if the considered difference does not exceed 5 PIP, otherwise it is considered large.

4 Recommended organization data structure

Students can choose organizing the data according to the following suggestion to perform assignment 1.

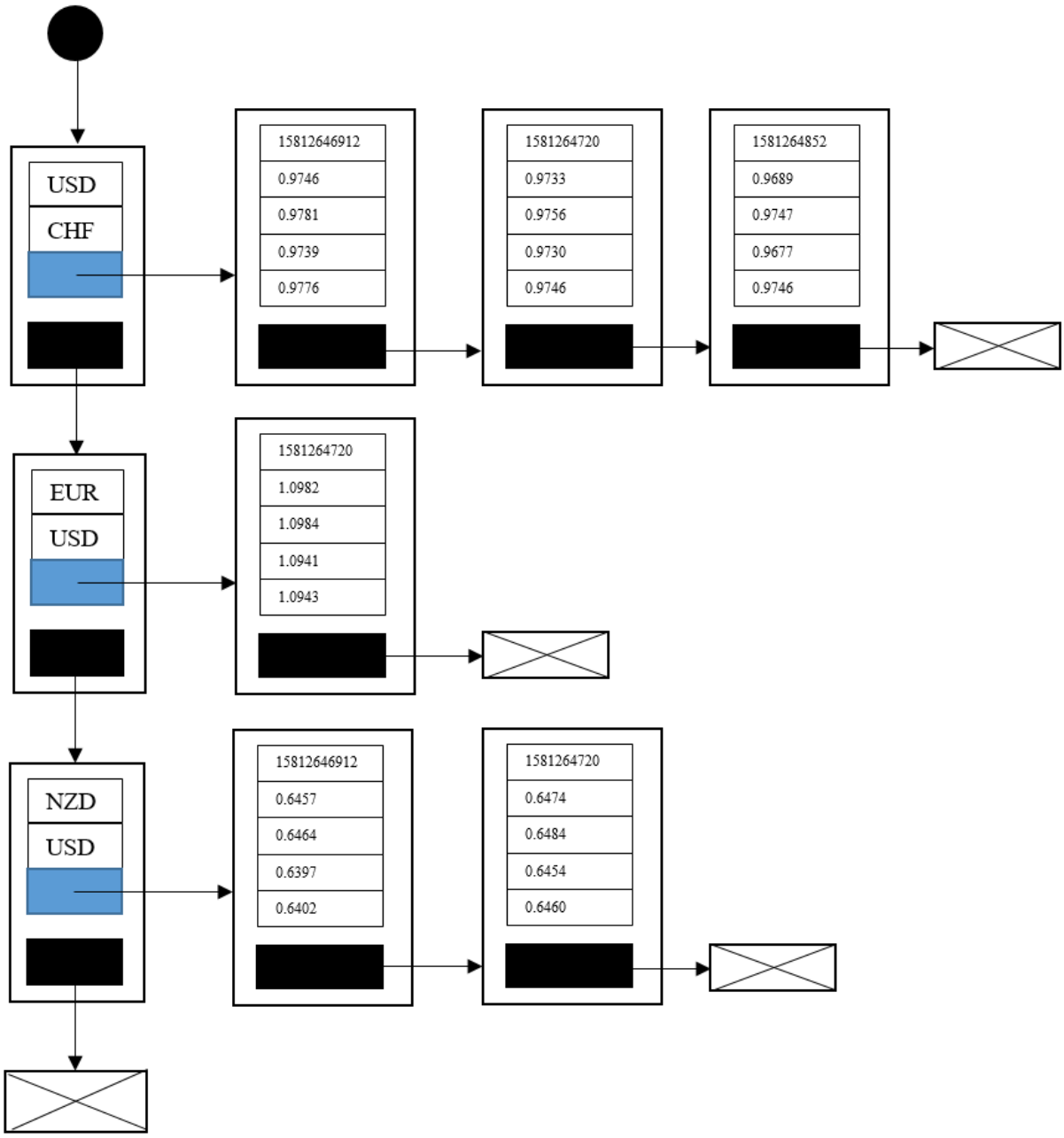

Figure 1: Represent data candlestick chart of each currency pair

To accomplish this task, a single linked list should be used to store currency pairs information, including:

- BC
- QC
- Link to a ordered link list of candles chronologically, corresponding to BC and QC.

(a) Information of a currency pair

(b) Information of a candle

Figure 2: The information are stored in the linked list

5 Submission

Students submit 2 files: **processData.h** and **processData.cpp** at "Assignment 1 Submission" in site "Cau truc du lieu và giai thuat (CO2003)_CC+CQ_(HK192)" on BKeL. The files must be in the plain form (NOT COMPRESSED), have the correct name and can be translated successfully when combined with the provided main.h and main.cpp files. Again, in the processData.h and processData.cpp files, students cannot add any #include other than the #include available in these files.

Deadline for submission is announced at the place to submit. Just after the dealine, the link to submit will be closed automatically so you cannot submit anything then. To avoid any risk, please upload your submission at least **one hour** before the dealine.

6 Plagiarism checking

The assignment must be DONE by yourself. Students will be considered cheating if:

- There is an unusual similarity between the source code of the submissions. In this case, ALL submissions are considered fraud. Therefore you must protect the source code of yourself

- Students do not understand the source code written by themselves, except for the initialized code. Students can refer to any resource, but make sure that you understand the meaning of all the commands you write. In case of not understanding the source code of the place where you refer, students are specifically warned that they should NOT use this source code; Instead, use what you have learned to write the program.
- Misleading submissions of other students on their personal accounts.

In the case of a conclusion of cheating, student score should be 0 for the entire course (not just only this assignment).

DO NOT ACCEPT ANY EXPLANATION AND NO EXCEPTIONS!

After each assignment is submitted, there will be a number of students who are randomly interviewed to prove that the assignment has been submitted by themselves.

7 Changelog

- Changes to the description below the table of instructions, the order distance changed from 5 PIP to 0.5 PIP.

References

[1] Tran Ngoc Bao Duy. *Assignment supporting document for Data structure and Algorithms - Foregin Exchange Market*, 2020.

——————————END——————————