Algoritmi e Calcolatori:
Lab 7: Templates and STL

Lab activities

- Practice with Templates.
- Practice with the STL Vector Containers.
- Recap all C++ lectures.

Exercise 1

Write a small program able to track down your monthly expenses. It requires a simple text file to store the transactions (one per line) according to the format:

```
<Date> <Description> <Amount>
```

where the `Amount` can be either a negative or positive number, i.e., respectively for when you buy something or you earn some money.
Let the user load this file when the program start: the filename could be a command-line argument. If the argument is missing the user is required to enter the filename at program startup. Let the user the possibility to add transactions, and update the file (without loosing any previous entered data). Before quitting, the program should show the balance.

Exercise 2

Starting from the previous exercise, expand its functionalities, as much as possible. Start by including a full year management.
In particular, the user needs the following operations:

- Add/Remove a transaction.
- Sort transactions by date (ascending and descending).
- Export the information into a different file.
- Show a report of all transactions.
- Show a summary report.

The report of all transaction should be in the following format:

```
| DATE        | DESCRIPTION     | INCOMES   | OUTCOMES  |
|-------------------------------------------------------|
| 25/03/2017  | Food            |           | 50€       |
| 27/03/2017  | Salary          | 1000€     |           |
|_____________|_________________|___________|___________|
```

The summary report shoud be in the following format:

```
| MONTHS  | INCOMES  | OUTCOMES  | BALANCE     |
|-------------------------------------------|
| 03/2017 | 1000€    | 830.57€   | +   169.43€ |
| 04/2017 | 1530€    | 1732.27€  | -   202.27€ |
|---------------------------------|-------------|
|---------------------------------| -    32.80€ |
|_________________________________|_____________|
```

To interact with the user, create a your own custom menu interface. Then exchange your (working) solution with someone else in order to test it and, possibly, gather some user-experience feedbacks.

Hint

Try to think on a base class, e.g., `Entry`, that manipulates a single transaction, having all public methods required for the task. Then develop other classes around that, so that they can manage the rest of the work. Remember, that all amounts, required a *currency*, which should be left open in the implementation, and they could be implemented using different *data types*.

Exercise 3

Improve the battlefield set of classes to allow the ability of tracking the position of each game element, i.e., soldiers, vehicles, and environmental objects, inside the battlefield map. In order to do so, try to define a way to track a *physical* map of the battlefield (e.g., a matrix of pixels, a matrix of tiles[1], etc.). Then, you can provide each class with the propers methods to track the position (and the size) of a single *element* with respect to the way you chose for build the map.
Once you find an effective solution, try to provide the battlefield with methods to sort each group of elements (e.g., all soldiers) by position (ascending, descending) and size. To do so, look for STL algorithms to reduce the size of your code. If necessary, convert your memory structures to STL containers.

[1] like in real video games `https://en.wikipedia.org/wiki/Tiled_rendering`