Scrabble Junior Game

WORK OVERVIEW

The objective of this practical work is to develop two programs: one to play a game called Scrabble Junior [1]; another to build game boards for that game (figure 1). The objective of the game is to cover the game board letters with matching letter tiles, and collect the most scoring chips by completing words. The playing rules can be found in reference [1]. The playing program must allow two to four players to play the game, detect the end of the game and announce the winner.

This is a group work; each group must have two students.

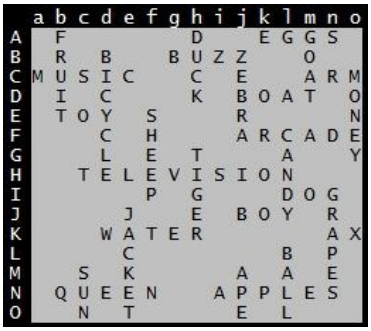

Figure 1. Initial view of a board.

LEARNING OBJECTIVES

The main objectives of this work are that students practice the development of programs, in **C ++ language**, following the **"object oriented" programming paradigm**, selecting data structures and algorithms suitable for the implementation of the game and taking advantage of the data structures and algorithms available in the **Standard Template Library (STL)** of C++. Other objectives like the use of files and the development of simple and robust program interfaces are also present.

PROGRAM SPECIFICATION

Note: please read the whole document, before starting the development.

The programs must execute in console mode, with a colored characters interface (when specified).

Program "Board Builder"

This is an auxiliary program to create game boards. The program must allow:

- To select the size of the game board, so that it is possible to play with boards of any size (maximum: 20x20).
- To create the game board, selecting a set of words, from the words available in a dictionary provided in the **WORDS.TXT** file, and positioning those words on the board, so that they intersect correctly. The position of the words must be specified by the user of the program, indicating the position of the first letter of the word and its orientation, horizontal or vertical. It is up to the program to verify that the words are valid, that they can be placed in the selected position (they fit inside the board) and that they intersect correctly.
- To save the contents of the board to a text file (example: **BOARD.TXT**).

The **format of the board file** must be the following (see an example in the annex):

- The first line contains the dimensions of the board (number_of_lines x number_of_columns).

- The next lines contain the list of words and their position (coordinates of the 1st letter) and orientation (horizontal or vertical) on the board. In each line, the sequence of the data must be: coordinates of the first letter, orientation and word (example: **Ca H MUSIC**)
- Optionally, a 2D view of the board can be added at the end of the file, with all the words placed, as in figure 1. This is not necessary for the development of the program, but facilitates the interpretation of the board contents when you open the file using a text editor.

An example is given in annex of a file describing the board in figure 1.

This program must be implemented using at least one class defined by the programmer: class **Board**, to represent a board.

Program "Scrabble Junior"

This program must allow **2 to 4 players** to play the game, detect the end of the game and announce the winner. Additional specifications are:

- The program must be implemented using some "classes" defined by the programmer, for example, to represent the board (**Board**), a player (**Player**), and the "bag" of letters (**Pool**); others may be useful or convenient. Note: taking into account the possibility of building your own boards, using the **Board Builder** program, the number of letter tiles may be different from those specified in reference [1]; the letter tiles in the pool must be the ones necessary to build the words in the board.
- The code must be structured in order to separate the definition of each class and the implementation of its methods, in files with the extension **.hpp** (or **.h**) and **.cpp**, respectively.
- The game board must be read from a text file, previously created. See the specification of the "**Board Builder**" program for the format of that file.
- The **number of players** and the **name of the board file** must be asked to the user at the beginning of the program.
- The game board must be shown with an aspect similar to that illustrated in figure 2. To specify the position of the letter to be played, users must use an **uppercase** letter to specify the **line** and a **lowercase** letter to specify the **column**. The range of the letters depends on the size of the board ('**A**' to '**O**' and '**a**' to '**o**', in the example).
- The program must prevent the playing of invalid letters (letters that are not in the player's hand) or at invalid positions (the played letter does not match with the letter in the chosen position).
- The way to indicate that a letter has been played on the board is to change the color of the letter on the screen (the letters in red, in figure 2).

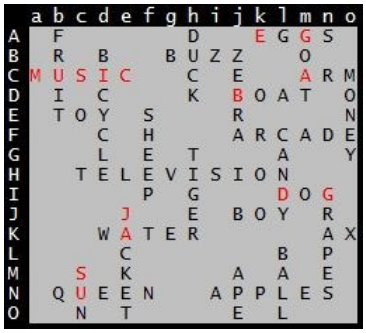

Figure 2. Board view after some moves of the players.

PROGRAM DEVELOPMENT

When writing the program code you should take into account the suggestions given in class, namely the ones concerning the following issues:

- Adequate choice of identifiers of types, variables and functions.
- Code commenting.
- Adequate choice of the data structures to represent the data manipulated by the program.
- Modular structure of the code.
- Adequate choice of function prototypes, namely, appropriate choice of parameters passed by value and by reference and adequate use of **const** qualifiers.

- Separation, as much as possible, of data processing from program input/output.
- Code robustness. Precautions should be taken in order to prevent the program to stop working due to incorrect input by the user, specially values outside the allowed ranges, and so on.
- Code efficiency.

WORK DEVELOPMENT AND SUBMISSION

- This work must be developed by **groups of two students**. The group members must belong to the same TP class (Portuguese "turma"). One student per group must inform the **code and name** of the **group members** to the teacher of the TP class, **until 24th/April**.
- **Create a folder** named **TxGyy**, in which **x** represents the class number and **yy** represents the number of the group (consult the list in Moodle), for example, **T5G01**, for the group number 1 from class ("turma") 5.
- In this folder, create a **ReadMe.txt** (in simple text format), indicating the development state of the two programs, that is, if all the objectives were accomplished or, otherwise, which ones were not achieved, and also which improvements were made, if any.
- Create 2 sub-folders: **BoardBuilder** and **ScrabbleJunior**.
- **Copy to each folder the source code of the corresponding program** (only the files with extension **.cpp** and **.hpp** or **.h**, if existing).
- **Compress** the content of the folder **TxGyy** into a file named **TxGyy.zip** and **upload** that file in the FEUP Moodle's page of the course. Alternative ways of delivering the work will not be accepted.
- **Important note**: if the code is not submitted in a **zip** file containing the **folder TxGyy** and the sub-folders **BoardBuilder** and **ScrabbleJunior**, as specified above, this practical work may not be graded.
- **Deadline** for submitting the work: **17th/May/2020 (at 23:55h)**.

REFERENCE

1. Scrabble Junior, https://www.hasbro.com/common/instruct/Scrabble_Junior.pdf

---

ANNEX

Example of a board file

```
15 x 15
Ak H EGGS
Bg H BUZZ
Ca H MUSIC
Cm H ARM
Dj H BOAT
Eb H TOY
Fj H ARCADE
Hc H TELEVISION
Il H DOG
Jj H BOY
Kd H WATER
Kn H AX
Nb H QUEEN
Ni H APPLES
Ab V FRUIT
Mc V SUN
Bd V BICYCLE
Je V JACKET
Ef V SHEEP
Ah V DUCK
Gh V TIGER
Bj V ZEBRA
Mj V APE
Fl V CANDY
Ll V BALL
Am V GOAT
In V GRAPES
Co V MONEY
```

(*Note: in this example, the 2D board is not represented at the end of the file*)