Programming Practices
CAA1 - 20232

Submission deadline: **17 / 3 / 2023**

Submission format and deadline

The CAA must be submitted before **March 17, 2023** at 23:59.

A single file must be submitted in **ZIP format**, containing:

- A document, in **PDF format,** with the answers to **exercises 1 and 3**. The first page of the document must contain **the name and surname** of the learner submitting the activity.
- Files **winegrower.h** and **winegrower.c** with the answer to **exercise 2** .

The submission must be made in the EC section of the theory classroom before the due date.

Learning Goals

- Review the basics of programming.
- Know how to interpret and follow the code of third parties.
- Know how to make small programs to solve problems based on an overview of the problem.
- Know how to specify an algorithm using pre and post conditions.
- Know how to include controls in programs to ensure that preconditions are met.

Correction criteria

Each exercise is associated with a weight over the total score of the activity. Both the correctness of the answers and their completeness will be valued.

- Failure to follow the **submission format**, both in terms of the **type and name of the files** and the content requested, will result in a **significant penalty** or a **0 rating of the activity**.
- In exercises where algorithmic language is required, the **format** must be respected.
- In the case of C language exercises, these **must correctly compile to be rated**. If they compile, it will be valued:
    - The code **works** as defined in the activity.
    - That the **style guides** are respected and that the code is properly **commented**.
    - Correctness of used code **structures**.

Notice

We take this opportunity to remind you that **it is totally forbidden to copy in the CAAs** of the course. It is understood that there may be work or communication between students during the performance of the activity, but the delivery of this must be individual and different from the rest. Deliveries will be analysed with plagiarism detection tools.

Therefore, submissions that contain an identical part with respect to submissions from other students will be considered copies and all those involved (regardless of the link between them) will fail the activity.

Citation guide: https://biblioteca.uoc.edu/en/contents/how-to-quote/index.html

Plagiarism monographic:

http://biblioteca.uoc.edu/en/biblioguides/biblioguide/Academic-plagiarism/

Observations

This CAA presents the project that will be developed during the different activities of the project, which has been simplified and adapted to the academic needs.

The following symbols are used in this document to refer to design and programming blocks:


Indicates that the code shown is in **algorithmic** language.


Indicates that the code shown is in **C language**.


Shows the **execution** of a program in C language.

Statement

In recent weeks there has been a widespread protest in Europe from the primary sector, both farmers and ranchers. One of the main complaints is the amount of bureaucracy required. Large producers are not exempt from it, but for small producers it means a great effort both in time and money if they require to outsource this documentation.

During this semester we will carry out a project related to a small part of the bureaucracy that a farmer faces. Wine being a highly known and exported Spanish product, we will focus on winegrowers associated with a Denomination of Origin (DO). In other words, a farmer who has a vineyard plantation and produces grapes that he sells to make wine. More specifically, the project will deal with the DOs located in Catalonia and that are regulated by the Wine Registry of Catalonia (RVC), and the procedures may be different in other regions of Spain.

The existing regulations in a DO require having a wine card, registering the plots (vineyards), declaring all production, etc. And our application must allow these procedures to be carried out.

In the CAA1 we will create a first version that will allow us to enter the basic data of a winegrower. Then, in future activities, we will expand the types and functions to obtain an application with more functionalities.

Nomenclature note: *The abbreviation DO will be used to refer to a Denomination of Origin, this being a Spanish nomenclature since in other countries other names are used for similar regulations. There is also a special category awarded to very few regions that certifies extra quality. In Catalonia the acronym DOQ is used (the Q is for "qualified") and the seal is only held by the "Priorat" and the equivalent in Spanish is the DOCa (the Ca is qualified) and the seal is only held by the "Rioja".*

Our application will connect to the final system, and therefore the input and output data will have to follow the agreed format. To facilitate this connection, it has been agreed that all applications will receive a file in CSV (Comma Separated Values) format for the input data and will generate the output in the same format. In this type of files, each line corresponds to a record, and the different fields of each record are separated with a known separator character. Specifically, in our case we will use the semicolon (;) as the separation character, and the formats will be:

Input data

The input to our application will be the **data of the people** registered as winegrowers, which will come in the following format:

"document;name;surname;phone;email;address;cp;birthday"

document : It refers to the identity document with which the person has registered in the system. This document uniquely identifies the person.
name : It refers to the name.
surname : It refers to surnames.
phone : It refers to the telephone. Digits and the character "+" are accepted in the first position. For example: +34699999999.
email : Refers to email.
address : Refers to the postal address.
cp: It refers to the zip code where you reside. The format is 5 digits, the first being a 0. For example: 08001
birthday : Refers to the date of birth in dd/mm/yyyy format (dd: day with two digits, mm: month with two digits, yyyy: year with four digits)

An example of a record would be:

"12345678N;Peter;Smith;+34699999999;peter.smith@example.com;My street, 25;08001;31/12/1980"

We will promptly receive **information about new winegrowers** who have registered in the registry, in the format:

"registrationDate;document;winegrowerId;DO;vineyardplotId;weight"

registrationDate : It refers to the date of registration of the winegrower, in dd/mm/yyyy format (dd: day with two digits, mm: month with two digits, yyyy: year with four digits).
document : It refers to the winegrower's identity document.
winegrowerId : It refers to the unique identifier of a winegrower. The format in Catalonia is a code of 14 characters that correspond to digits.
DO : It refers to the code of the DO associated with the winegrower. It is a 2-character code, with values such as: PR (Priorat), EM (Empordà), TA (Terra Alta),

PE (Penedés), AL (Alella), MO (Montsant), TA (Tarragona), CB (Conca de Barberá), CS (Costers del Segre), etc.
vineyardplotId: It refers to the registered vineyard code of the winegrower for the associated DO and is a 13-character string.
weight: Total annual production capacity in kilograms of grapes per hectare, of the winegrower in the registered vineyard. Two figures are always used for decimals.

Some log examples would be:

"01/12/2023;12345678N;01234567890123;TA;TA12345678001;3900.80"
"15/01/2024;87654321B;56789012345678;PR;PR87654321001;1184.80"

***Note**:* Fields containing text do not have line breaks or the ";" character. They also do not contain accents or umlauts.

Output data

The output of the application will be the messages to send. These messages will be sent by another application, so a CSV file will have to be generated with the necessary information. As this sending application will be used for different purposes, the messages have a generic format and the data is passed in predefined fields. Each line will be a message in the following format:

"email;when;who;what"

email : Refers to the email address of the recipient of the message.
when : Date of registration of the winegrower in the registry. The format will be "dd/mm/yyyy" (dd: day with two digits, mm: month with two digits, yyyy: year with four digits).
who : Identifier of the winegrower, in the previously specified format.
what : DO in the previously specified format of 2 characters associated with the winegrower.

An example of a record would be:

"peter.smith@example.com;15/01/2024;56789012345678;PR"

Given that people's data and outgoing messages are used in various applications, as part of the project they have provided us with the definition of the following types (those that you need to solve the exercises must be translated into C language):

```
const
      MAX_PEOPLE: integer = 100;
end const

type
      tDate = record
            day: integer;
            month: integer;
            year: integer;
      end record

      tPerson = record
            document: string;
            name: string;
            surname: string;
            phone: string;
            email: string;
            address: string;
            cp: string;
            birthday: tDate;
      end record

      tPeople = record
            elems: vector [MAX_PEOPLE] of tPerson;
            count: integer;
      end record

      tMessage = record
            email: string;
            when: tDate;
            who: string;
            what: string;
      end record
end type
```

Exercise 1: Defining data types [25%]

From the description of the input data of the statement, define in **algorithmic language**:

- A **tWinegrower** type that can save the information of a winegrower with his document, his winegrower identifier, the date of registration in the registry, the associated DO code and the vineyard code and the annual production of grapes per hectare for that vineyard and DO.

- A **tWinegrowerData** type that can save the information of all the winegrowers who have registered in the registry. You can assume that we will save a table with a maximum of 200 records.

Note: You can define the additional types that you consider appropriate. You can also use the data types of the statement.

Exercise 2: Table manipulation [50%]

Starting from the data structures defined in exercise 1 and the structures and methods detailed at the end of this statement, implement the following methods (actions or functions) in C language (use only the **winegrower.h** and **winegrower.c** files to declare and implement the methods):

a) **winegrowerData_init:** Given a structure of type **tWinegrowerData**, initialize it correctly by getting an empty structure.

b) **winegrowerData_len:** Given a structure of type **tWinegrowerData**, returns the number of registered winegrowers, stored in this structure.

c) **winegrower_parse:** Given a structure of type **tWinegrower** and a CSV file entry (**tCSVEntry**) with the data of a winegrower, initialize the structure with this data.

d) **winegrowerData_add**: Given a structure of type **tWinegrowerData** and the code of a winegrower (**tWinegrower**), register this winegrower by adding it to the list of registered winegrowers (**tWinegrowerData**). If the winegrower code already exists with the same vineyard code and DO, it will not be added but the registration date of that entry will be updated and the total kilograms will be accumulated in its annual production capacity per hectare. If the winegrower code already exists in the table but is associated with another vineyard and/or DO code, the registration is not carried out. Note: *A winegrower only has one vineyard and one DO associated with him*.

e) **winegrowerData_get:** Given a structure of type **tWinegrowerData**, a position, and an output string, writes the winegrower information at the given position of **tWinegrowerData** to the string. The character string will follow the same format as the input data. The position may not be valid. An example output would be (see the C **sprintf** method):

"01/12/2023;12345678N;01234567890123;TA;TA12345678001;3900.80"

f) **winegrowerData_del:** Given a structure of type **tWinegrowerData** and a winegrower code, remove this winegrower from the list. If no winegrower exists with that code, then the method does nothing.

You must use the main routine provided in the **main.c** file without any modification. This routine executes the following tasks (and displays the stored data after each step):

1. Add 3 new records
2. Add 2 existing records
3. Delete 1 existing records
4. Delete 1 non-existent records

To consider that the program works correctly, the final result on the screen should be the following:

```
0;01/12/2023;12345678N;01234567890123;TA;TA12345678001;1184.00
1;10/12/2023;55755010C;23456789012345;MO;MO55755010001;2500.10
2;15/01/2024;87654321B;56789012345678;PR;PR87654321001;4800.50


0;01/12/2023;12345678N;01234567890123;TA;TA12345678001;1184.00
1;10/12/2023;55755010C;23456789012345;MO;MO55755010001;2500.10
2;15/01/2024;87654321B;56789012345678;PR;PR87654321001;5300.50


0;01/12/2023;12345678N;01234567890123;TA;TA12345678001;1184.00
1;15/01/2024;87654321B;56789012345678;PR;PR87654321001;5300.50


0;01/12/2023;12345678N;01234567890123;TA;TA12345678001;1184.00
1;15/01/2024;87654321B;56789012345678;PR;PR87654321001;5300.50
```

Note: To carry out this exercise, in the files **csv.h**, **csv.c**, **winegrower.h** and **winegrower.c** you will find the declaration and implementation of the following methods:

tCSVEntry	Data type that saves a line of a CSV file.
csv_numFields	Given a tCSVEntry structure, returns the number of fields it contains.
csv_getAsInteger	Given a tCSVEntry structure and the position of a field (0 indicates the first position), returns its value as an integer.
csv_getAsString	Given a tCSVEntry structure and the position of a field (0 indicates the first position), returns its value as a string.

csv_getAsReal	Given a tCSVEntry structure and the position of a field (0 indicates the first position), returns its value as real.
tDate	Structure that allows saving information about a date.
date_parse	Given a structure of type tDate and a string with a date (in the input data format), initialize the tDate structure with the date.
date_equals	Compares two tDate structures and indicates whether they contain the same value or not.

Exercise 3: Formal specification [25%]

Defines the declaration (**not the implementation**) of the **winegrowerData_init** and **winegrowerData_add** methods from the previous exercise in algorithmic language.

a) Add pre and post conditions to the declaration of these methods in **algorithmic language.**

b) Add the necessary *asserts* to the C language code of the previous exercise to ensure that the pre-conditions specified in these methods are met.