



PROGRAMMING TECHNIQUES DEGREE IN APPLIED MATHEMATICS AND COMPUTING Carlos III University of Madrid ACADEMIC YEAR 2022-2023	 
---	--

FINAL PROJECT

Rules:

- The project must be implemented by groups of 2 members.
- Write your names and NIAs at the beginning of the program.
- The project has a value of 25% from the total mark of the course.
- The project must be submitted using Aula Global, NOT by email.
- Only one member of the group must submit the project.
- The project must be submitted using only one script **project.cpp**.
- You are not allowed to change the names of the classes and the functions expressed in the project description.
- IF YOU NEED, you can implement and use more functions, but you must justify them.

Project Statement:

A C++ program is needed to implement an algorithm that solves the problem of TRAINS MANAGEMENT in a passenger transport company. The program should consist of the following data types:

A train is identified by the following information:

- departure date where a train departs from a specific station. To handle the date attribute, you can use the class Date with the format dd:mm:yyyy, in order to make it easy there is no need to use time. We suppose that each train has only one trip per day.
- departure station
- arrival station.
- distance in KM between departure and arrival stations.
- wagons, where each train consists of a group of wagons. To save information about wagons, you can use any appropriate container except arrays.

A wagon is identified by the following information:

- an integer number which represents the number of a wagon (1, 2, 3,).

- seats number, which is the number of seats in each wagon. Wagons have different numbers of seats. The maximum number of passengers in a wagon must be less or equal the number of seats of that wagon.
- IDs (national identity documents) to store the ID of all the passengers who occupied the seats in a specific wagon. For IDs you can use any appropriate container except arrays.

A passenger is identified by the following information:

- ID (national identity document) is an identification key to avoid duplication of passengers.
- passenger's name.
- passenger's address.
- passenger's age.
- passenger's gender (M, F).
- baggage weight (in KG) where a passenger is allowed to carry on-board the train. As a maximum, a passenger is allowed to carry 25kg onboard the train.
- trips that are done by passengers. To save information about all the trips, you can use any appropriate container except arrays

For a trip, we need to store the following information:

- date of the current trip.
- departure station.
- arrival station.
- money paid for the current trip.
- seat number where a passenger has been assigned for the current trip.
- wagon number where a passenger has been assigned for the current trip.
- distance of the current trip in KM.

The program must contain at least the following functions:

- **readInitialData:** This function receives a reference to an empty object (list) of trains and a reference to an empty object (map) of passengers, where the key is the ID of a passenger, and the value is the object of the passenger itself. The function is used to fill in the previous two objects with appropriate information. You have to provide information about at least two trains that depart from the same station, e.g., Madrid. A train consists of at least two wagons. Each wagon contains at least five passengers. This function is executed at the beginning of the program to modify the previous two objects. It does not return anything. You can invent all the information about the trains (wagons, passengers, baggage, etc). The function reads the information from external files. You can use one file or several files.

- **mainMenu:** This function does not receive anything. It shows on the screen the following options and returns the option selected by the user. The menu must control that the user cannot choose another option.
 1. **Add new passenger's trip to a train**
 2. **Remove a passenger's trip from a train**
 3. **Show trips of a passenger**
 4. **Show list of passengers in a specific train**
 5. **Show an alphabetically ordered list of passengers in all the trains**
 6. **Show passengers of cities**
 7. **End program**

When the user selects one of the previous options, one of the following corresponding function will be executed.

1. **addNewPassengerTrip:** This function receives the references of the two objects; trains and passengers. It asks the user to introduce the date of the trip, departure station, arrival station, and the ID of the new passenger. It checks if the introduced passenger does not exist in the object of passengers. The function inserts that passenger into the object of passengers. In addition, it checks if the given passenger exists on the train, then the function sends a message indicating that the passenger exists on the given train. In this case, the function returns the wagon number and the seat number of the passenger. If the new passenger is not found on the given train, the function adds the introduced passenger to the first unoccupied seat, and updates the object of trips of the given passenger.
2. **removePassengerTrip:** This function is used to remove a passenger's trip from a specific train. It asks the user to introduce the date of the trip, departure station, arrival station, and the ID of the passenger. If the given passenger is not found on that train, the function returns a message to the user. If not, the function removes the introduced passenger's trip and returns a message that the trip has been removed. When a trip is removed, you have to update the trips of that passenger.
3. **showTripsOfPassenger:** This function is used to show all the trips of a given passenger. It asks the user to introduce the ID of a passenger. If the given passenger is not found, the function returns a message to the user. If the passenger is found, it shows a list of all the trips made by that passenger.
4. **showListOfPassengers:** This function is used to show on the screen a list of the passengers on a specific train. It asks the user to introduce a code for a train. If the given train is not found, the function returns a message to the user. If not, the function shows a list of all the passengers (ID and name) with the wagon number and the seat number.

5. **showOrderedListOfPassengers:** This function is used to return an alphabetically ordered list (according to the names) of the passengers on all the trains. The list must contain the date of the departure, departure station, arrival station, wagon number and seat number. The list must be printed on the screen.
6. **showPassengersCities:** The function is used to return a map container, where the key represents a city name, and the value represents a list of the passengers' names who visited that city. Print the content of the container on the screen.
7. **endProgram:** This function is used to print all the information of the trains and passengers into an external file 'Output.txt'. There are no style restrictions on displaying the information, but understandability will be considered.

Note: Your program should fulfil the following requirements:

- Implement the date types that are needed to handle the previous information.
- All the attributes of the classes must be defined as private attributes.
- All the classes must contain default constructors and parametrized constructors.
- The program must have the necessary error control to ensure its stability.