Caso_di_Studio-Carpool

Data Structure Documentation

Booking_travel_t Struct Reference

`#include <Carpool.h>`

Data Fields

- char **departure_destination** [**MAX_LENGHT_STRINGS**]
- char **arrival_destination** [**MAX_LENGHT_STRINGS**]
- **Date_t departure_date**
- **Time_t departure_time**
- unsigned short **number_seats**

Detailed Description

This user-defined type is used in order to book a travel.

Field Documentation

char Booking_travel_t::arrival_destination[MAX_LENGHT_STRINGS]

This member is used to store the arrival destination of the travel that the user wants to book

Date_t Booking_travel_t::departure_date

This member is used to store the departure date of the travel that the user wants to book

char Booking_travel_t::departure_destination[MAX_LENGHT_STRINGS]

This member is used to store the departure destination of the travel that the user wants to book

Time_t Booking_travel_t::departure_time

This member is used to store the departure time of the travel that the user wants to book

unsigned short Booking_travel_t::number_seats

This member is used to store the number of seats that the user needs in order to book the travel

---

The documentation for this struct was generated from the following file:

- C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/**Carpool.h**

Date_t Struct Reference

```
#include <Date.h>
```

Data Fields

- unsigned short **year**
- **Month_t month**
- unsigned short **day**

Detailed Description

This user-defined type is used in order to manage the dates.

Field Documentation

unsigned short Date_t::day

This member is used to store the day of the date

Month_t Date_t::month

This member is used to store the month of the date

unsigned short Date_t::year

This member is used to store the year of the date

The documentation for this struct was generated from the following file:

- C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/**Date.h**

Driver_t Struct Reference

```
#include <Carpool.h>
```

Data Fields

- int **id**
- char **name** [**MAX_LENGHT_STRINGS**]
- char **surname** [**MAX_LENGHT_STRINGS**]
- char **email** [**MAX_LENGHT_EMAIL**]
- char **password** [**MAX_LENGHT_STRINGS**]
- char **phone_number** [**MAX_LENGHT_PHONE_NUMBER**]
- **Date_t birthday**
- **Gender_t gender**
- **Rating_t driving_capacity**
- **Rating_t comfort_capacity**
- **Rating_t average_rating**
- bool **deleted**

Detailed Description

This user-defined type is used in order to manage drivers.

Field Documentation

Rating_t Driver_t::average_rating

This member is used to store the driver's average rating

Date_t Driver_t::birthday

This member is used to store the driver's birthday

Rating_t Driver_t::comfort_capacity

This member is used to store the driver's comfort capacity

bool Driver_t::deleted

This member is used to know if the driver is deleted, if this member is true, means that the driver is deleted

Rating_t Driver_t::driving_capacity

This member is used to store the driver's driving capacity

char Driver_t::email[MAX_LENGHT_EMAIL]

This member is used to store the driver's email

Gender_t Driver_t::gender

This member is used to store the driver's gender

int Driver_t::id

This member is used to store the driver's ID

char Driver_t::name[MAX_LENGHT_STRINGS]

This member is used to store the driver's name

char Driver_t::password[MAX_LENGHT_STRINGS]

This member is used to store the driver's password

char Driver_t::phone_number[MAX_LENGHT_PHONE_NUMBER]

This member is used to store the driver's phone number

char Driver_t::surname[MAX_LENGHT_STRINGS]

This member is used to store the driver's surname

The documentation for this struct was generated from the following file:

- C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/**Carpool.h**

Rating_file_t Struct Reference

`#include <Carpool.h>`

Data Fields

- int **id_driver**
- bool **option_rating**
- **Rating_t rating**

Detailed Description

This user-defined type is used in order to save rating into a file.

Field Documentation

int Rating_file_t::id_driver

This member is used to store the ID of the driver that has been evalutated

bool Rating_file_t::option_rating

This member is used to know which evalutation the user wants to do, if "option rating" is true it means that rating refers to driver capacity, otherwise it refers to comfort capacity

Rating_t Rating_file_t::rating

This member is used to store the evalutation that the user has made

---

The documentation for this struct was generated from the following file:

- C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/**Carpool.h**

Time_t Struct Reference

```
#include <Date.h>
```

Data Fields

- unsigned short **hour**
- unsigned short **minute**

Detailed Description

This user-defined type is used in order to manage the times.

Field Documentation

unsigned short Time_t::hour

This member is used to store the hour of the time

unsigned short Time_t::minute

This member is used to store the minute of the time

The documentation for this struct was generated from the following file:

- C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/**Date.h**

Travel_t Struct Reference

`#include <Carpool.h>`

Data Fields

- int **id**
- int **id_driver**
- char **departure_destination** [**MAX_LENGHT_STRINGS**]
- char **arrival_destination** [**MAX_LENGHT_STRINGS**]
- char **additional_notes** [**MAX_LENGHT_ADDITIONAL_NOTES**]
- **Date_t departure_date**
- **Time_t departure_time**
- double **price**
- unsigned short **total_seats**
- unsigned short **free_seats**
- bool **deleted**

Detailed Description

This user-defined type is used in order to manage travels.

Field Documentation

char Travel_t::additional_notes[MAX_LENGHT_ADDITIONAL_NOTES]

This member is used to store the travel's additional notes

char Travel_t::arrival_destination[MAX_LENGHT_STRINGS]

This member is used to store the travel's arrival destination

bool Travel_t::deleted

This member is used to know if the travel is deleted, if this member is true, means that the travel is deleted

Date_t Travel_t::departure_date

This member is used to store the travel's departure date

char Travel_t::departure_destination[MAX_LENGHT_STRINGS]

This member is used to store the travel's departure destination

Time_t Travel_t::departure_time

This member is used to store the travel's departure time

unsigned short Travel_t::free_seats

This member is used to store the veicle's free seats

int Travel_t::id

This member is used to store the travel's ID

int Travel_t::id_driver

This member is used to store the ID of the driver that will offer the travel

double Travel_t::price

This member is used to store the travel's price

unsigned short Travel_t::total_seats

This member is used to store the veicle's total seats (It must include the driver's seat)

The documentation for this struct was generated from the following file:

- C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/**Carpool.h**

File Documentation

Mainpage.md File Reference

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/Carpool.c File Reference

This file is the implementation file of **Carpool.h**.

```
#include "Carpool.h"
```

Functions

- const char * **readGender** (const **Gender_t** ***gender**)
- const char * **readRating** (const **Rating_t** *rating)
- void **setWord** (char word[], const char printf_value[])
- void **setEmail** (char **email**[])
- void **setPassword** (char **password**[])
- void **setPhoneNumber** (char **phone_number**[])
- void **setAdditionalNotes** (char **additional_notes**[])
- void **setPrice** (double ***price**)
- void **setNumberInput** (int *input, const int min, const int max, const char printf_value_input[], const char printf_value_error[])
- void **resetDriver** (**Driver_t** *driver)
- void **setDriver** (**Driver_t** *driver, const int *id)
- void **readDriver** (const **Driver_t** *driver)
- bool **isIdDriverEqual** (const **Driver_t** *driver, const int *id)
- void **showMemberDriver** (void)
- void **showSortKeyDriver** (void)
- void **resetTravel** (**Travel_t** *travel)
- void **setTravel** (**Travel_t** *travel, const int *id, const char path_file_driver[])
- void **readTravel** (const **Travel_t** *travel, const char path_driver_file[])
- bool **isIdTravelEqual** (const **Travel_t** *travel, const int *id)
- void **showMemberTravel** (void)
- void **showSortKeyTravel** (void)
- void **resetBookingTravel** (**Booking_travel_t** *booking_travel)
- **File_status_t addStruct** (const char path_file_driver[], const char path_file_travel[], const int *id, bool select_struct)
- **File_status_t editStruct** (const char path_file_driver[], const char path_file_travel[], bool select_struct)
- **File_status_t deleteStruct** (const char path_file_driver[], const char path_file_travel[], bool select_struct)
- **File_status_t showAllStructs** (const char path_file_driver[], const char path_file_travel[], bool select_struct)
- bool **bookTravel** (const char path_file_driver[], const char path_file_travel[])
- **File_status_t manageRating** (const char path_file_driver[], const char path_file_rating[])
- **File_status_t evaluateDriver** (const char path_file_driver[], const char path_file_rating[])
- **File_status_t updateID** (const char path_file[], const long int offset, int *id)
- long int **getIndexUser** (const char path_file_driver[], const char path_file_travel[], const char printf_value_input[], const char printf_value_error[], bool select_struct)
- long int **getIndex** (const char path_file[], const int *id, bool select_struct)
- double **setSort** (const char path_file[], long int start, long int end, bool select_struct)
- void **mergeSort** (const char path_file[], long int start, long int end, bool select_struct, int key_sort)
- void **mergeDriver** (const char path_file[], long int start, long int middle, long int end, int key_sort)
- void **mergeTravel** (const char path_file[], long int start, long int middle, long int end, int key_sort)

Detailed Description

This file is the implementation file of **Carpool.h**.

Author

Vincenzo Susso

Date

2019 Sep 10

Version

1.0

Function Documentation

addStruct (const char *path_file_driver*[], const char *path_file_travel*[], const int * *id*, bool *select_struct*)

This function is used to set a struct between **Driver_t** and **Travel_t** and save them into a file.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file.
id	is the unique ID of driver or travel.
select_struct	is used to set **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then a driver will be added to the system, if select_struct is equal to "TRAVEL" then a travel will be added to the system.

Returns

2 if the struct is added to the system, otherwise this function will return 0.

bookTravel (const char *path_file_driver*[], const char *path_file_travel*[])

This function is used in order to book a travel.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file

Returns

true if a travel has been booked, otherwise this function will return false

deleteStruct (const char *path_file_driver*[], const char *path_file_travel*[], bool *select_struct*)

This function is used to delete a struct between **Driver_t** and **Travel_t**.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file
select_struct	is used to delete **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then a driver will be deleted, if select_struct is equal to "TRAVEL" then a travel will be deleted.

Returns

2 if the struct has been deleted, otherwise this function will return 0.

editStruct (const char *path_file_driver*[], const char *path_file_travel*[], bool *select_struct*)

This function is used to edit a struct between **Driver_t** and **Travel_t** and save them into a file.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file
select_struct	is used to edit **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then a driver will be edited and added to the system, if select_struct is equal to "TRAVEL" then a travel will be edited and added to the system.

Returns

2 if the struct has been edited and added to the system, otherwise this function will return 0.

evaluateDriver (const char *path_file_driver*[], const char *path_file_rating*[])

This function is used in order to allow users to enter evaluations to the drivers.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_rating	is the relative path of the rating's file.

Returns

1 if the user has evaluate driver, otherwise this function will return 0.

getIndex (const char *path_file*[], const int * *id*, bool *select_struct*)

This function is used to return the index of the ID that is passed by pointer.

Parameters

path_file	is the relative path of the driver's file.
id	is used to search the index of the struct.
select_struct	is used to get index of **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the driver's index will be searched, if select_struct is equal to "TRAVEL" then the travel's index will be searched.

Returns

the index of the struct if it has been found, otherwise the function will return 0.

getIndexUser (const char *path_file_driver*[], const char *path_file_travel*[], const char *printf_value_input*[], const char *printf_value_error*[], bool *select_struct*)

This function is used to return the index of the ID that is entered by the user using keyboard.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file.
printf_value_input	is used to ask to the user to enter the ID of the struct he needs.
printf_value_error	is used to report to the user if there was an error during the entering of the ID that he needs.
select_struct	is used to get index of **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the driver's index will be searched, if select_struct is equal to "TRAVEL" then the travel's index will be searched.

Returns

the index of the struct if it has been found, otherwise the function will return 0.

isIdDriverEqual (const Driver_t * *driver*, const int * *id*)

This function is used to compare the driver's ID passed by pointer with the other id passed by pointer. P.S: if the driver is deleted, the function will not compares the ID.

Parameters

driver	is used as first member of the comparision.
id	is used as the second member of the comparision.

Returns

true if the driver's ID is equal to id, otherwise the function will return false.

isIdTravelEqual (const Travel_t * *travel*, const int * *id*)

This function is used to compare the travel's ID passed by pointer with the other id passed by pointer. P.S: if the travel is deleted, the function will not compares the ID.

Parameters

travel	is used as first member of the comparision.
id	is used as second member of the comparision.

Returns

true if the travel's ID is equal to id, otherwise the function will return false.

manageRating (const char *path_file_driver*[], const char *path_file_rating*[])

This function is used in order to assign the evalutations to the drivers.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_rating	is the relative path of the rating's file.

Returns

1 if evalutations have been assigned to the drivers, otherwise this function will return 0.

mergeDriver (const char *path_file*[], long int *start*, long int *middle*, long int *end*, int *key_sort*)

This procedure is used to merge the driver's records.

Parameters

path_file	is the relative path of the driver's file that the user wants to sort.
start	is the offset of the first record of the file.
middle	is the offset of the medium driver's record.
end	is the offset of the last record of the file.
key_sort	is used to indicate the key sort of the sorting.

mergeSort (const char *path_file*[], long int *start*, long int *end*, bool *select_struct*, int *key_sort*)

This procedure is used to split the records.

Parameters

path_file	is the relative path of the file that the user wants to sort.
start	is the offset of the first record of the file.
end	is the offset of the last record of the file.
select_struct	is used to sort **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the drivers will be sorted, if select_struct is equal to "TRAVEL" then the travels will be sorted.
key_sort	is used to indicate the key sort of the sorting.

mergeTravel (const char *path_file*[], long int *start*, long int *middle*, long int *end*, int *key_sort*)

This procedure is used to merge the travel's records.

Parameters

path_file	is the relative path of the travel's file that the user wants to sort.
start	is the offset of the first record of the file.
middle	is the offset of the medium driver's record.
end	is the offset of the last record of the file.
key_sort	is used to indicate the key sort of the sorting.

readDriver (const Driver_t * *driver*)

This procedure prints every members of the drivers passed by pointer. P.S: if the driver is deleted, the procedure will not read the driver.

Parameters

driver	is printed.

*** readGender (const Gender_t * *gender*)**

This function returns the letteral output using the pointer *gender as a index of array's string. For more information, please visit here: `https://stackoverflow.com/questions/1496313/returning-c-string-from-a-function`

Parameters

gender	is used to return a letteral output.

Returns

"Male" if (*gender) is equal to 0, "Female" if (*gender) is equal to 1, otherwise It will return "Custom".

*** readRating (const Rating_t * *rating*)**

This function returns the letteral output using the pointer *rating as a index of array's string. For more information, please visit here: `https://stackoverflow.com/questions/1496313/returning-c-string-from-a-function`

Parameters

rating	is used to return a letteral output.

Returns

"None" if (*rating) is equal to 0, " " if (*rating) is equal to 1, "**" if (*rating) is equal to 2, "***" if (*rating) is equal to 3, "****" if (*rating) is equal to 4 and "*****" if (*rating) is equal to 5.

readTravel (const Travel_t * *travel*, const char *path_driver_file*[])

This procedure prints every members of the travel passed by pointer. P.S: if the travel is deleted, the procedure will not read the travel.

Parameters

travel	is printed.
path_driver_file	is used to print information of the driver that will offer the travel.

resetBookingTravel (Booking_travel_t * *booking_travel*)

This procedure reset the booking_travel passed by pointer assigning invalid values to all the booking_travel's members.

Parameters

booking_travel	is resetted by the procedure.

resetDriver (Driver_t * *driver*)

This procedure reset the driver passed by pointer assigning invalid values to all the driver's members.

Parameters

driver	is resetted by the procedure.

resetTravel (Travel_t * *travel*)

This procedure reset the travel passed by pointer assigning invalid values to all the travel's members.

Parameters

travel	is resetted by the procedure.

setAdditionalNotes (char *additional_notes*[])

This procedure is used to set a valid value to the additional notes passed by pointer. The additional notes should not be void strings and can contains spaces.

Parameters

additional_notes	is set to a valid additional notes.

setDriver (Driver_t * *driver*, const int * *id*)

This procedure set valid value to every members of the driver passed by pointer.

Parameters

driver	is used to set valid value to every members.
id	is the unique id of the driver.

setEmail (char *email*[])

This procedure is used to set a valid value to the email passed by pointer. A valid email has the following format "localpart@domain".

Parameters

email	is set to a valid email.

setNumberInput (int * *input*, const int *min*, const int *max*, const char *printf_value_input*[], const char *printf_value_error*[])

This procedure is used to set a valid value to the input passed by pointer. A valid value is made only of digits.

Parameters

input	is set to a valid number.
min	is the minimun valid number.
max	is the maximum valid number.
printf_value_input	is used to ask to the user what he should enter.
printf_value_error	is used to report to the user if there was an error during the entering of the number.

setPassword (char *password*[])

This procedure is used to set a valid value to the password passed by pointer. The password should contains at least one uppercase character and one digit.

Parameters

password	is set to a valid password.

setPhoneNumber (char *phone_number*[])

This procedure is used to set a valid value to the number phone passed by pointer. A valid number phone has the followig format "+xxx xxxxxxxxxxxx".

Parameters

phone_number	is set to a valid number phone.

setPrice (double * *price*)

This procedure is used to set a valid value to the price passed by pointer.

Parameters

price	is set to a valid price.

setSort (const char *path_file*[], long int *start*, long int *end*, bool *select_struct*)

This function is used to ask to the user to enter the key_sort.

Parameters

path_file	is the relative path of the file that the user wants to sort.

start	is the offset of the first record of the file.
end	is the offset of the last record of the file.
select_struct	is used to sort **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the drivers will be sorted, if select_struct is equal to "TRAVEL" then the travels will be sorted.

Returns

the time that the sorting has spent, otherwise this function will return 0.

void setTravel (Travel_t * *travel*, const int * *id*, const char *path_file_driver*[])

setWord (char *word*[], const char *printf_value*[])

This procedure is used to set a valid value to the word passed by pointer. A valid word is made of only latin characters and it is not void.

Parameters

word	is set to a valid string.
printf_value	is used to ask to the user what he should enter.

showAllStructs (const char *path_file_driver*[], const char *path_file_travel*[], bool *select_struct*)

This function is used to show all records of **Driver_t** or **Travel_t**.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file
select_struct	is used to read all records of **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then all drivers will be read (except the deleted ones), if select_struct is equal to "TRAVEL" then all travels will be read (except the deleted ones).

Returns

1 if all the records has been read, otherwise this function will return 0.

showMemberDriver (void)

This procedure is used to show all the driver's member. This procedure is used during the editing of driver's member.

showMemberTravel (void)

This procedure is used to show all the travel's member. This procedure is used during the editing of travel's member.

showSortKeyDriver (void)

This procedure is used to show all the driver's sort-key.

showSortKeyTravel (void)

This procedure is used to show all the travel's sort-key.

updateID (const char *path_file*[], const long int *offset*, int * *id*)

This function is used in order to update the ID passed by pointer and save its into the file.

Parameters

path_file	is the relative path where IDs are stored.
offset	can be set to "OFFSET_ID_DRIVER" in order to update the unique ID of the drivers, otherwise offset can be set to "OFFSET_ID_TRAVEL" in order to update the unique ID of the travels.
id	is the unique identifier that will be updated.

Returns

1 if the ID was updated, otherwise the function will return 0

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/Carpool.h File Reference

This library was created in order to manage drivers and travels.

```
#include <stdbool.h>
#include <stdio.h>
#include <limits.h>
#include "Date.h"
#include "Utilities.h"
#include "File.h"
```

Data Structures

- struct **Driver_t**
- struct **Travel_t**
- struct **Rating_file_t**
- struct **Booking_travel_t**

Macros

- #define **DRIVER** true
- #define **TRAVEL** false
- #define **MAX_LENGHT_STRINGS** 20
- #define **MIN_LENGHT_STRINGS** 2
- #define **MAX_LENGHT_EMAIL** 40
- #define **MIN_LENGHT_PASSWORD** 8
- #define **MIN_LENGHT_PHONE_NUMBER** 8
- #define **MAX_LENGHT_PHONE_NUMBER** 18
- #define **MAX_LENGHT_ADDITIONAL_NOTES** 40
- #define **MAX_LENGHT_NUMBER_INPUT** 8
- #define **MIN_YEAR_BIRTHDAY** 1915
- #define **MAX_YEAR_BIRTHDAY** 2001
- #define **MIN_YEAR_TRAVEL** 2019
- #define **MAX_YEAR_TRAVEL** 2050
- #define **MIN_PRICE** 0.01
- #define **MAX_PRICE** 9999.99
- #define **DOLLAR_STRING** "$"
- #define **MIN_NUMBER_TOTAL_SEATS** 2
- #define **MAX_NUMBER_TOTAL_SEATS** 9
- #define **MIN_NUMBER_FREE_SEATS** 0
- #define **MAX_NUMBER_FREE_SEATS** 8
- #define **LENGHT_ARRAY_GENDER** 3
- #define **READ_GENDER_MALE** "Male"
- #define **READ_GENDER_FEMALE** "Female"
- #define **READ_GENDER_CUSTOM** "Custom"
- #define **LENGHT_ARRAY_RATING** 6
- #define **READ_RATING_NONE** "None"
- #define **READ_RATING_ONE_STAR** "*"
- #define **READ_RATING_TWO_STAR** "**"
- #define **READ_RATING_THREE_STAR** "***"
- #define **READ_RATING_FOUR_STAR** "****"
- #define **READ_RATING_FIVE_STAR** "*****"
- #define **ALREADY_SORTED** 1
- #define **OFFSET_ID_DRIVER** 0
- #define **OFFSET_ID_TRAVEL** 1

- #define **MERGE_TEMP_FILE_PATH** "../Files/TempSort.dat"
- #define **BOOK_TRAVEL_TEMP_FILE_PATH** "../Files/TempBook.dat"

Enumerations

- enum **Rating_t** { **none**, **one_star**, **two_star**, **three_star**, **four_star**, **five_star** }
- enum **Gender_t** { **male**, **female**, **custom** }
- enum **Driver_members_t** { **id_driver** = -1, **name**, **surname**, **email**, **password**, **phone_number**, **birthday**, **gender**, **deleted_driver** }
- enum **Driver_sort_key** { **inc_id_driver**, **dec_id_driver**, **inc_name**, **dec_name**, **inc_surname**, **dec_surname**, **inc_birthday**, **dec_birthday**, **inc_gender**, **dec_gender**, **inc_driving_capacity**, **dec_driving_capacity**, **inc_comfort_capacity**, **dec_comfort_capacity**, **inc_average_rating**, **dec_average_rating** }
- enum **Travel_members_t** { **id_travel** = -2, **id_driver_**, **departure_destination**, **arrival_destination**, **departure_date**, **departure_time**, **total_seats**, **free_seats**, **price**, **additional_notes**, **deleted_travel** }
- enum **Travel_sort_key** { **inc_id_travel**, **dec_id_travel**, **inc_departure_destination**, **dec_departure_destination**, **inc_arrival_destination**, **dec_arrival_destination**, **inc_departure_date**, **dec_departure_date**, **inc_price**, **dec_price**, **inc_total_seats**, **dec_total_seats**, **inc_free_seats**, **dec_free_seats** }

Functions

- const char * **readGender** (const **Gender_t** ***gender**)
- const char * **readRating** (const **Rating_t** *rating)
- void **setWord** (char word[], const char printf_value[])
- void **setEmail** (char **email**[])
- void **setPassword** (char **password**[])
- void **setPhoneNumber** (char **phone_number**[])
- void **setAdditionalNotes** (char **additional_notes**[])
- void **setPrice** (double ***price**)
- void **setNumberInput** (int *input, const int min, const int max, const char printf_value_input[], const char printf_value_error[])
- void **resetDriver** (**Driver_t** *driver)
- void **setDriver** (**Driver_t** *driver, const int *id)
- void **readDriver** (const **Driver_t** *driver)
- bool **isIdDriverEqual** (const **Driver_t** *driver, const int *id)
- void **showMemberDriver** (void)
- void **showSortKeyDriver** (void)
- void **resetTravel** (**Travel_t** *travel)
- void **setTravel** (**Travel_t** *travel, const int *id, const char path_file_driver[])
- void **readTravel** (const **Travel_t** *travel, const char path_driver_file[])
- bool **isIdTravelEqual** (const **Travel_t** *travel, const int *id)
- void **showMemberTravel** (void)
- void **showSortKeyTravel** (void)
- void **resetBookingTravel** (**Booking_travel_t** *booking_travel)
- **File_status_t addStruct** (const char path_file_driver[], const char path_file_travel[], const int *id, bool select_struct)
- **File_status_t editStruct** (const char path_file_driver[], const char path_file_travel[], bool select_struct)
- **File_status_t deleteStruct** (const char path_file_driver[], const char path_file_travel[], bool select_struct)
- **File_status_t showAllStructs** (const char path_file_driver[], const char path_file_travel[], bool select_struct)
- bool **bookTravel** (const char path_file_driver[], const char path_file_travel[])
- **File_status_t manageRating** (const char path_file_driver[], const char path_file_rating[])
- **File_status_t evaluateDriver** (const char path_file_driver[], const char path_file_rating[])
- **File_status_t updateID** (const char path_file[], const long int offset, int *id)

- long int **getIndexUser** (const char path_file_driver[], const char path_file_travel[], const char printf_value_input[], const char printf_value_error[], bool select_struct)
- long int **getIndex** (const char path_file[], const int *id, bool select_struct)
- double **setSort** (const char path_file[], long int start, long int end, bool select_struct)
- void **mergeSort** (const char path_file[], long int start, long int end, bool select_struct, int key_sort)
- void **mergeDriver** (const char path_file[], long int start, long int middle, long int end, int key_sort)
- void **mergeTravel** (const char path_file[], long int start, long int middle, long int end, int key_sort)

Detailed Description

This library was created in order to manage drivers and travels.

Author

Vincenzo Susso

Date

2019 Sep 10

Version

1.0 This library was developed to create, edit and delete drivers and travels, furthermore this library allows to save drivers and travels into files.

Macro Definition Documentation

#define ALREADY_SORTED 1

This integer is used to see if the number of record is one, this case means that the records are already sorted.

#define BOOK_TRAVEL_TEMP_FILE_PATH "../Files/TempBook.dat"

This string is used to indicates the relative path of a temporary file used during the booking of a travel.

#define DOLLAR_STRING "$"

This string is used in order to indicates the currency of the travel's price.

#define DRIVER true

This boolean is used to indicates that the a struct of **Driver_t** will be modified into the procedure and functions.

#define LENGHT_ARRAY_GENDER 3

This integer is used to indicates the lenght of the array that is used to converts the numeral output of gender to letteral output.

#define LENGHT_ARRAY_RATING 6

This integer is used to indicates the lenght of the array that is used to converts the numeral output of rating to letteral output.

#define MAX_LENGHT_ADDITIONAL_NOTES 40

This integer is used to indicates the maximum lenght of additional notes.

#define MAX_LENGHT_EMAIL 40

This integer is used to indicates the maximum lenght of emails.

#define MAX_LENGHT_NUMBER_INPUT 8

This integer is used to indicates lenght of string that is used to take a number as input.

#define MAX_LENGHT_PHONE_NUMBER 18

This integer is used to indicates the maximum lenght of phone_number.

#define MAX_LENGHT_STRINGS 20

This integer is used to indicates the maximum lenght of strings.

#define MAX_NUMBER_FREE_SEATS 8

This integer is used to indicates the maximum number of free seats in a veicle.

#define MAX_NUMBER_TOTAL_SEATS 9

This integer is used to indicates the maximum number of total seats in a veicle (It include the driver's seat).

#define MAX_PRICE 9999.99

This double is used to indicates the maximum travel's price.

#define MAX_YEAR_BIRTHDAY 2001

This integer is used to indicates the maximum valid year to be a driver.

#define MAX_YEAR_TRAVEL 2050

This integer is used to indicates the maximum valid year to create a travel.

#define MERGE_TEMP_FILE_PATH "../Files/TempSort.dat"

This string is used to indicates the relative path of a temporary file used during the sorting.

#define MIN_LENGHT_PASSWORD 8

This integer is used to indicates the minimum lenght of passwords.

#define MIN_LENGHT_PHONE_NUMBER 8

This integer is used to indicates the minimum lenght of phone number.

#define MIN_LENGHT_STRINGS 2

This integer is used to indicates the minimum lenght of strings.

#define MIN_NUMBER_FREE_SEATS 0

This integer is used to indicates the mimimum number of free seats in a veicle.

#define MIN_NUMBER_TOTAL_SEATS 2

This integer is used to indicates the minimum number of total seats in a veicle (It include the driver's seat).

#define MIN_PRICE 0.01

This double is used to indicates the minimum travel's price.

#define MIN_YEAR_BIRTHDAY 1915

This integer is used to indicates the minimum valid year to be a driver.

#define MIN_YEAR_TRAVEL 2019

This integer is used to indicates the minimum valid year to create a travel.

#define OFFSET_ID_DRIVER 0

This integer is used to update the driver's ID.

#define OFFSET_ID_TRAVEL 1

This integer is used to update the travel's ID.

#define READ_GENDER_CUSTOM "Custom"

This string is used as string that will be shown instead of numeral output.

#define READ_GENDER_FEMALE "Female"

This string is used as string that will be shown instead of numeral output.

#define READ_GENDER_MALE "Male"

This string is used as string that will be shown instead of numeral output.

#define READ_RATING_FIVE_STAR "***"**

This string is used as string that will be shown instead of numeral output.

#define READ_RATING_FOUR_STAR "**"**

This string is used as string that will be shown instead of numeral output.

#define READ_RATING_NONE "None"

This string is used as string that will be shown instead of numeral output.

#define READ_RATING_ONE_STAR "*"

This string is used as string that will be shown instead of numeral output.

#define READ_RATING_THREE_STAR "*"**

This string is used as string that will be shown instead of numeral output.

#define READ_RATING_TWO_STAR ""**

This string is used as string that will be shown instead of numeral output.

#define TRAVEL false

This boolean is used to indicates that the a struct of **Travel_t** will be modified into the procedure and functions.

Enumeration Type Documentation

enum Driver_members_t

This user-defined type is used in order to define the member of the struct **Driver_t**, this user-defined type was created in order to edit the member of the struct **Driver_t**.

Enumerator:

XE "id_driverCarpool.h"XE "Carpool.hid_driver"id_driver	This member is used to indicates the driver's ID
XE	This member is used to indicate the driver's name

"nameCarpool.h"XE "Carpool.hname"name	
XE "surnameCarpool.h"XE "Carpool.hsurname"surname	This member is used to indicate the driver's surname
XE "emailCarpool.h"XE "Carpool.hemail"email	This member is used to indicate the driver's email
XE "passwordCarpool.h"XE "Carpool.hpassword"password	This member is used to indicate the driver's password
XE "phone_numberCarpool.h"XE "Carpool.hphone_number"phone_number	This member is used to indicate the driver's phone_number
XE "birthdayCarpool.h"XE "Carpool.hbirthday"birthday	This member is used to indicate the driver's birthday
XE "genderCarpool.h"XE "Carpool.hgender"gender	This member is used to indicate the driver's gender
XE "deleted_driverCarpool.h"XE "Carpool.hdeleted_driver"deleted_driver	This member is used to indicate the driver's deletion

enum Driver_sort_key

This user-defined type is used in order to sort the drivers using several sort-key.

This user-defined type is used in order to sort the travels using several sort-key.

Enumerator:

XE "inc_id_driverCarpool.h"XE "Carpool.hinc_id_driver"inc_id_driver	This member is used to sort drivers using increasing ID as sorting-key
XE "dec_id_driverCarpool.h"XE "Carpool.hdec_id_driver"dec_id_driver	This member is used to sort drivers using decreasing ID as sorting-key
XE "inc_nameCarpool.h"XE "Carpool.hinc_name"inc_name	This member is used to sort drivers using increasing driver's name as sorting-key
XE "dec_nameCarpool.h"XE "Carpool.hdec_name"de	This member is used to sort drivers using decreasing driver's name as sorting-key

c_name	
XE "inc_surnameCarpool.h "XE "Carpool.hinc_surname "inc_surname	This member is used to sort drivers using increasing driver's surname as sorting-key
XE "dec_surnameCarpool.h "XE "Carpool.hdec_surname "dec_surname	This member is used to sort drivers using decreasing driver's surname as sorting-key
XE "inc_birthdayCarpool.h" XE "Carpool.hinc_birthday" inc_birthday	This member is used to sort drivers using increasing driver's birthday as sorting-key
XE "dec_birthdayCarpool.h "XE "Carpool.hdec_birthday "dec_birthday	This member is used to sort drivers using decreasing driver's birthday as sorting-key
XE "inc_genderCarpool.h" XE "Carpool.hinc_gender"inc_gender	This member is used to sort drivers using increasing driver's gender as sorting-key
XE "dec_genderCarpool.h" XE "Carpool.hdec_gender" dec_gender	This member is used to sort drivers using decreasing driver's gender as sorting-key
XE "inc_driving_capacityCarpool.h"XE "Carpool.hinc_driving_capacity"inc_driving_capacity	This member is used to sort drivers using increasing driver's driving capacity as sorting-key
XE "dec_driving_capacityCarpool.h"XE "Carpool.hdec_driving_capacity"dec_driving_capacity	This member is used to sort drivers using decreasing driver's driving capacity as sorting-key
XE "inc_comfort_capacityCarpool.h"XE "Carpool.hinc_comfort_capacity"inc_comfort_capacity	This member is used to sort drivers using increasing driver's comfort capacity as sorting-key
XE "dec_comfort_capacityCarpool.h"XE "Carpool.hdec_comfort_capacity"dec_comfort_capacity	This member is used to sort drivers using decreasing driver's comfort capacity as sorting-key
XE "inc_average_ratingCarpool.h"XE "Carpool.hinc_average_	This member is used to sort drivers using increasing driver's average rating as sorting-key

rating"inc_average_rating	
XE "dec_average_ratingCarpool.h"XE "Carpool.hdec_average_rating"dec_average_rating	This member is used to sort drivers using decreasing driver's average rating as sorting-key

enum Gender_t

This user-defined type is used to know the gender of the driver, this user-defined type was also created in order to improve the readability.

Enumerator:

XE "maleCarpool.h"XE "Carpool.hmale"male	This member is used to indicate that the driver's gender is male
XE "femaleCarpool.h"XE "Carpool.hfemale"female	This member is used to indicate that the driver's gender is female
XE "customCarpool.h"XE "Carpool.hcustom"custom	This member is used to indicate that the driver's gender is custom

enum Rating_t

This user-defined type is used to evalutate the driver's capacity, this user-defined type was also created in order to improve the readability.

Enumerator:

XE "noneCarpool.h"XE "Carpool.hnone"none	none This member is used when driver has no rating
XE "one_starCarpool.h"XE "Carpool.hone_star"one_star	one_star This member is used to assign one star rating to the driver
XE "two_starCarpool.h"XE "Carpool.htwo_star"two_star	two_star This member is used to assign two star rating to the driver
XE "three_starCarpool.h"XE "Carpool.hthree_star"three_star	three_star This member is used to assign three star rating to the driver
XE "four_starCarpool.h"XE "Carpool.hfour_star"four_star	four_star This member is used to assign four star rating to the driver
XE "five_starCarpool.h"XE "Carpool.hfive_star"five_star	five_star This member is used to assign five star rating to the driver

enum Travel_members_t

This user-defined type is used in order to define the member of the struct **Travel_t**, this user-defined type was created in order to edit the member of the struct **Travel_t**.

Enumerator:

XE "id_travelCarpool.h"XE "Carpool.hid_travel"id_travel	This member is used to indicates the travel's ID
XE "id_driver_Carpool.h"XE "Carpool.hid_driver_"id_driver_	This member is used to indicates the ID of the driver that will offer the travel
XE "departure_destinationCarpool.h"XE "Carpool.hdeparture_destination"departure_destination	This member is used to indicates the travel's departure destination
XE "arrival_destinationCarpool.h"XE "Carpool.harrival_destination"arrival_destination	This member is used to indicates the travel's arrival destination
XE "departure_dateCarpool.h"XE "Carpool.hdeparture_date"departure_date	This member is used to indicates the travel's departure date
XE "departure_timeCarpool.h"XE "Carpool.hdeparture_time"departure_time	This member is used to indicates the travel's departure time
XE "total_seatsCarpool.h"XE "Carpool.htotal_seats"total_seats	This member is used to indicates the veicle's total seats
XE "free_seatsCarpool.h"XE "Carpool.hfree_seats"free_seats	This member is used to indicates the veicle's free seats
XE "priceCarpool.h"XE "Carpool.hprice"price	This member is used to indicates the travel's price
XE "additional_notesCarpool.h"XE "Carpool.hadditional_notes"additional_notes	This member is used to indicates the travel's additional notes
XE "deleted_travelCarpool.h"XE "Carpool.hdeleted_travel"deleted_travel	This member is used to indicate the travel's deletion

enum Travel_sort_key

Enumerator:

XE "inc_id_travelCarpool.h "XE "Carpool.hinc_id_travel "inc_id_travel	This member is used to sort travels using increasing ID as sorting-key
XE "dec_id_travelCarpool.h "XE "Carpool.hdec_id_travel "dec_id_travel	This member is used to sort travels using decreasing ID as sorting-key
XE "inc_departure_destinationCarpool.h"XE "Carpool.hinc_departure_destination"inc_departure_destination	This member is used to sort travels using increasing departure destination as sorting-key
XE "dec_departure_destinationCarpool.h"XE "Carpool.hdec_departure_destination"dec_departure_destination	This member is used to sort travels using decreasing departure destination as sorting-key
XE "inc_arrival_destinationCarpool.h"XE "Carpool.hinc_arrival_destination"inc_arrival_destination	This member is used to sort travels using increasing arrival destination as sorting-key
XE "dec_arrival_destinationCarpool.h"XE "Carpool.hdec_arrival_destination"dec_arrival_destination	This member is used to sort travels using decreasing arrival destination as sorting-key
XE "inc_departure_dateCarpool.h"XE "Carpool.hinc_departure_date"inc_departure_date	This member is used to sort travels using increasing departure date as sorting-key
XE "dec_departure_dateCarpool.h"XE "Carpool.hdec_departure_date"dec_departure_date	This member is used to sort travels using decreasing departure date as sorting-key
XE "inc_priceCarpool.h"XE "Carpool.hinc_price"inc_price	This member is used to sort travels using increasing price as sorting-key
XE "dec_priceCarpool.h"XE "Carpool.hdec_price"dec_price	This member is used to sort travels using decreasing price as sorting-key

XE "inc_total_seatsCarpool.h"XE "Carpool.hinc_total_seats"inc_total_seats	This member is used to sort travels using increasing veicle's total seats as sorting-key
XE "dec_total_seatsCarpool.h"XE "Carpool.hdec_total_seats"dec_total_seats	This member is used to sort travels using decreasing veicle's total seats as sorting-key
XE "inc_free_seatsCarpool.h"XE "Carpool.hinc_free_seats"inc_free_seats	This member is used to sort travels using increasing veicle's free seats as sorting-key
XE "dec_free_seatsCarpool.h"XE "Carpool.hdec_free_seats"dec_free_seats	This member is used to sort travels using decreasing veicle's free seats as sorting-key

Function Documentation

File_status_t addStruct (const char *path_file_driver*[], const char *path_file_travel*[], const int * *id*, bool *select_struct*)

This function is used to set a struct between **Driver_t** and **Travel_t** and save them into a file.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file.
id	is the unique ID of driver or travel.
select_struct	is used to set **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then a driver will be added to the system, if select_struct is equal to "TRAVEL" then a travel will be added to the system.

Returns

2 if the struct is added to the system, otherwise this function will return 0.

bool bookTravel (const char *path_file_driver*[], const char *path_file_travel*[])

This function is used in order to book a travel.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file

Returns

true if a travel has been booked, otherwise this function will return false

File_status_t deleteStruct (const char *path_file_driver*[], const char *path_file_travel*[], bool *select_struct*)

This function is used to delete a struct between **Driver_t** and **Travel_t**.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file
select_struct	is used to delete **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then a driver will be deleted, if select_struct is equal to "TRAVEL" then a travel will be deleted.

Returns

2 if the struct has been deleted, otherwise this function will return 0.

File_status_t editStruct (const char *path_file_driver*[], const char *path_file_travel*[], bool *select_struct*)

This function is used to edit a struct between **Driver_t** and **Travel_t** and save them into a file.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file
select_struct	is used to edit **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then a driver will be edited and added to the system, if select_struct is equal to "TRAVEL" then a travel will be edited and added to the system.

Returns

2 if the struct has been edited and added to the system, otherwise this function will return 0.

File_status_t evaluateDriver (const char *path_file_driver*[], const char *path_file_rating*[])

This function is used in order to allow users to enter evaluations to the drivers.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_rating	is the relative path of the rating's file.

Returns

1 if the user has evaluate driver, otherwise this function will return 0.

long int getIndex (const char *path_file*[], const int * *id*, bool *select_struct*)

This function is used to return the index of the ID that is passed by pointer.

Parameters

path_file	is the relative path of the driver's file.
id	is used to search the index of the struct.
select_struct	is used to get index of **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the driver's index will be searched, if select_struct is equal to "TRAVEL" then the travel's index will be searched.

Returns

the index of the struct if it has been found, otherwise the function will return 0.

long int getIndexUser (const char *path_file_driver*[], const char *path_file_travel*[], const char *printf_value_input*[], const char *printf_value_error*[], bool *select_struct*)

This function is used to return the index of the ID that is entered by the user using keyboard.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file.
printf_value_input	is used to ask to the user to enter the ID of the struct he needs.
printf_value_error	is used to report to the user if there was an error during the entering of the ID that he needs.
select_struct	is used to get index of **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the driver's index will be searched, if select_struct is equal to "TRAVEL" then the travel's index will be searched.

Returns

the index of the struct if it has been found, otherwise the function will return 0.

bool isIdDriverEqual (const Driver_t * *driver*, const int * *id*)

This function is used to compare the driver's ID passed by pointer with the other id passed by pointer. P.S: if the driver is deleted, the function will not compares the ID.

Parameters

driver	is used as first member of the comparision.
id	is used as the second member of the comparision.

Returns

true if the driver's ID is equal to id, otherwise the function will return false.

bool isIdTravelEqual (const Travel_t * *travel*, const int * *id*)

This function is used to compare the travel's ID passed by pointer with the other id passed by pointer. P.S: if the travel is deleted, the function will not compares the ID.

Parameters

travel	is used as first member of the comparision.
id	is used as second member of the comparision.

Returns

true if the travel's ID is equal to id, otherwise the function will return false.

File_status_t manageRating (const char *path_file_driver*[], const char *path_file_rating*[])

This function is used in order to assign the evalutations to the drivers.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_rating	is the relative path of the rating's file.

Returns

1 if evalutations have been assigned to the drivers, otherwise this function will return 0.

void mergeDriver (const char *path_file*[], long int *start*, long int *middle*, long int *end*, int *key_sort*)

This procedure is used to merge the driver's records.

Parameters

path_file	is the relative path of the driver's file that the user wants to sort.
start	is the offset of the first record of the file.
middle	is the offset of the medium driver's record.
end	is the offset of the last record of the file.
key_sort	is used to indicate the key sort of the sorting.

void mergeSort (const char *path_file*[], long int *start*, long int *end*, bool *select_struct*, int *key_sort*)

This procedure is used to split the records.

Parameters

path_file	is the relative path of the file that the user wants to sort.
start	is the offset of the first record of the file.
end	is the offset of the last record of the file.
select_struct	is used to sort **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the drivers will be sorted, if select_struct is equal to "TRAVEL" then the travels will be sorted.
key_sort	is used to indicate the key sort of the sorting.

void mergeTravel (const char *path_file*[], long int *start*, long int *middle*, long int *end*, int *key_sort*)

This procedure is used to merge the travel's records.

Parameters

path_file	is the relative path of the travel's file that the user wants to sort.
start	is the offset of the first record of the file.
middle	is the offset of the medium driver's record.
end	is the offset of the last record of the file.
key_sort	is used to indicate the key sort of the sorting.

void readDriver (const Driver_t * *driver*)

This procedure prints every members of the drivers passed by pointer. P.S: if the driver is deleted, the procedure will not read the driver.

Parameters

driver	is printed.

const char* readGender (const Gender_t * *gender*)

This function returns the letteral output using the pointer *gender as a index of array's string. For more information, please visit here: `https://stackoverflow.com/questions/1496313/returning-c-string-from-a-function`

Parameters

gender	is used to return a letteral output.

Returns

"Male" if (*gender) is equal to 0, "Female" if (*gender) is equal to 1, otherwise It will return "Custom".

const char* readRating (const Rating_t * *rating*)

This function returns the letteral output using the pointer *rating as a index of array's string. For more information, please visit here: `https://stackoverflow.com/questions/1496313/returning-c-string-from-a-function`

Parameters

rating	is used to return a letteral output.

Returns

"None" if (*rating) is equal to 0, " " if (*rating) is equal to 1, "**" if (*rating) is equal to 2, "***" if (*rating) is equal to 3, "****" if (*rating) is equal to 4 and "*****" if (*rating) is equal to 5.

void readTravel (const Travel_t * *travel*, const char *path_driver_file*[])

This procedure prints every members of the travel passed by pointer. P.S: if the travel is deleted, the procedure will not read the travel.

Parameters

travel	is printed.
path_driver_file	is used to print information of the driver that will offer the travel.

void resetBookingTravel (Booking_travel_t * *booking_travel*)

This procedure reset the booking_travel passed by pointer assigning invalid values to all the booking_travel's members.

Parameters

booking_travel	is resetted by the procedure.

void resetDriver (Driver_t * *driver*)

This procedure reset the driver passed by pointer assigning invalid values to all the driver's members.

Parameters

driver	is resetted by the procedure.

void resetTravel (Travel_t * *travel*)

This procedure reset the travel passed by pointer assigning invalid values to all the travel's members.

Parameters

travel	is resetted by the procedure.

void setAdditionalNotes (char *additional_notes*[])

This procedure is used to set a valid value to the additional notes passed by pointer. The additional notes should not be void strings and can contains spaces.

Parameters

additional_notes	is set to a valid additional notes.

void setDriver (Driver_t * *driver*, const int * *id*)

This procedure set valid value to every members of the driver passed by pointer.

Parameters

driver	is used to set valid value to every members.
id	is the unique id of the driver.

void setEmail (char *email*[])

This procedure is used to set a valid value to the email passed by pointer. A valid email has the following format "localpart@domain".

Parameters

email	is set to a valid email.

void setNumberInput (int * *input*, const int *min*, const int *max*, const char *printf_value_input*[], const char *printf_value_error*[])

This procedure is used to set a valid value to the input passed by pointer. A valid value is made only of digits.

Parameters

input	is set to a valid number.
min	is the minimun valid number.
max	is the maximum valid number.
printf_value_input	is used to ask to the user what he should enter.
printf_value_error	is used to report to the user if there was an error during the entering of the number.

void setPassword (char *password*[])

This procedure is used to set a valid value to the password passed by pointer. The password should contains at least one uppercase character and one digit.

Parameters

password	is set to a valid password.

void setPhoneNumber (char *phone_number*[])

This procedure is used to set a valid value to the number phone passed by pointer. A valid number phone has the followig format "+xxx xxxxxxxxxxxx".

Parameters

phone_number	is set to a valid number phone.

void setPrice (double * *price*)

This procedure is used to set a valid value to the price passed by pointer.

Parameters

price	is set to a valid price.

double setSort (const char *path_file*[], long int *start*, long int *end*, bool *select_struct*)

This function is used to ask to the user to enter the key_sort.

Parameters

path_file	is the relative path of the file that the user wants to sort.
start	is the offset of the first record of the file.
end	is the offset of the last record of the file.
select_struct	is used to sort **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then the drivers will be sorted, if select_struct is equal to "TRAVEL" then the travels will be sorted.

Returns

the time that the sorting has spent, otherwise this function will return 0.

void setTravel (Travel_t * *travel*, const int * *id*, const char *path_file_driver*[])

void setWord (char *word*[], const char *printf_value*[])

This procedure is used to set a valid value to the word passed by pointer. A valid word is made of only latin characters and it is not void.

Parameters

word	is set to a valid string.
printf_value	is used to ask to the user what he should enter.

File_status_t showAllStructs (const char *path_file_driver*[], const char *path_file_travel*[], bool *select_struct*)

This function is used to show all records of **Driver_t** or **Travel_t**.

Parameters

path_file_driver	is the relative path of the driver's file.
path_file_travel	is the relative path of the travel's file
select_struct	is used to read all records of **Driver_t** or **Travel_t**, if select_struct is equal to "DRIVER" then all drivers will be read (except the deleted ones), if select_struct is equal to "TRAVEL" then all travels will be read (except the deleted ones).

Returns

1 if all the records has been read, otherwise this function will return 0.

void showMemberDriver (void)

This procedure is used to show all the driver's member. This procedure is used during the editing of driver's member.

void showMemberTravel (void)

This procedure is used to show all the travel's member. This procedure is used during the editing of travel's member.

void showSortKeyDriver (void)

This procedure is used to show all the driver's sort-key.

void showSortKeyTravel (void)

This procedure is used to show all the travel's sort-key.

File_status_t updateID (const char *path_file*[], const long int *offset*, int * *id*)

This function is used in order to update the ID passed by pointer and save its into the file.

Parameters

path_file	is the relative path where IDs are stored.
offset	can be set to "OFFSET_ID_DRIVER" in order to update the unique ID of the drivers, otherwise offset can be set to "OFFSET_ID_TRAVEL" in order to update the unique ID of the travels.
id	is the unique identifier that will be updated.

Returns

1 if the ID was updated, otherwise the function will return 0

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/Date.c File Reference

This file is the implementation file of **Date.h**.

```
#include "Date.h"
```

Functions

- bool **isLeapYear** (const unsigned short year)
- bool **isValidDate** (const **Date_t** *date, const unsigned short min_year, const unsigned short max_year)
- void **resetDate** (**Date_t** *date)
- void **setDate** (**Date_t** *date, const unsigned short min_year, const unsigned short max_year, const char printf_value[])
- bool **isValidTime** (const **Time_t** *time)
- void **resetTime** (**Time_t** *time)
- void **setTime** (**Time_t** *time, const char printf_value[])
- **Date_order_t cmpDate** (const **Date_t** *first_date, const **Date_t** *second_date)
- **Date_order_t cmpTime** (const **Time_t** *first_time, const **Time_t** *second_time)

Detailed Description

This file is the implementation file of **Date.h**.

Author

Vincenzo Susso

Date

2019 Sep 09

Version

1.0

Function Documentation

cmpDate (const Date_t * *first_date*, const Date_t * *second_date*)

This functions is used to compare two dates.

Parameters

first_date	passed by pointer is used as a date to compare.
second_date	passed by pointer is used as a date to compare.

Returns

-1 if the first date is older than the second one, 0 if the first date is equal to the second one, 1 if the first date is later than the second one.

cmpTime (const Time_t * *first_time*, const Time_t * *second_time*)

Parameters

first_time	passed by pointer is used as a time to compare.
second_time	passed by pointer is used as a time to compare.

Returns

-1 if the first time is older than the second one, 0 if the first time is equal to the second one, 1 if the first time is later than the second one.

isLeapYear (const unsigned short *year*)

In the Gregorian calendar, every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400.

This function checks if the year is a leap year.

Parameters

year	is checked in order to see if it is a leap year.

Returns

true if the year is a leap year, otherwise It will return false.

isValidDate (const Date_t * *date*, const unsigned short *min_year*, const unsigned short *max_year*)

This function checks if the date that has been passed by pointer is a valid date and It is included between min_year and max_year.

Parameters

date	is checked in order to see if it is valid.
min_year	is the older valid year.
max_year	is the later valid year.

Returns

true if the date is a valid date, otherwise It will return false.

isValidTime (const Time_t * *time*)

This function checks if the time that has been passed by pointer is a valid time.

Parameters

time	is checked in order to see if it is valid.

Returns

true if the time is a valid time, otherwise It will return false.

resetDate (Date_t * *date*)

This procedure reset the date passed by pointer assigning invalid values to the date.

Parameters

date	is resetted by the procedure.

resetTime (Time_t * *time*)

This procedure reset the time passed by pointer assigning invalid values to the time.

Parameters

time	is resetted by the procedure

setDate (Date_t * *date*, const unsigned short *min_year*, const unsigned short *max_year*, const char *printf_value*[])

This procedure sets a valid date to the date passed by pointer, the date must be included between min_year and max_year.

Parameters

date	is used to assigns a valid value.
min_year	is the older valid year.
max_year	is the later valid_year.
printf_value	says to the users what they should enter.

setTime (Time_t * *time*, const char *printf_value*[])

This procedure sets a valid time to the time passed by pointer.

Parameters

time	is used to assigns a valid value.
printf_value	says to the users what they should enter.

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/Date.h File Reference

This library was created in order to provide some procedures and functions that are used to manage dates and times.

```
#include <stdbool.h>
#include <string.h>
#include "Utilities.h"
```

Data Structures

- struct **Date_t**
- struct **Time_t**

Macros

- #define **MIN_DAY** 1
- #define **MAX_DAY** 31
- #define **MAX_DAY_FEBRUARY** 29
- #define **CENTURY_YEAR** 100
- #define **CENTURY_LEAP_YEAR** 400
- #define **LEAP_YEAR** 4
- #define **MIN_HOUR** 0
- #define **MAX_HOUR** 23
- #define **MIN_MINUTE** 0
- #define **MAX_MINUTE** 59
- #define **DATE_DELIMITER** "/"
- #define **TIME_DELIMITER** ":"
- #define **MAX_LENGHT_DATE_STRING_INPUT** 11
- #define **MAX_LENGHT_TIME_STRING_INPUT** 6

Enumerations

- enum **Month_t** { **january** = 1, **february**, **march**, **april**, **may**, **june**, **july**, **august**, **september**, **october**, **november**, **december** }
- enum **Date_order_t** { **older** = -1, **equal**, **later** }

Functions

- bool **isLeapYear** (const unsigned short year)
- bool **isValidDate** (const **Date_t** *date, const unsigned short min_year, const unsigned short max_year)
- void **resetDate** (**Date_t** *date)
- void **setDate** (**Date_t** *date, const unsigned short min_year, const unsigned short max_year, const char printf_value[])
- bool **isValidTime** (const **Time_t** *time)
- void **resetTime** (**Time_t** *time)
- void **setTime** (**Time_t** *time, const char printf_value[])
- **Date_order_t cmpDate** (const **Date_t** *first_date, const **Date_t** *second_date)
- **Date_order_t cmpTime** (const **Time_t** *first_time, const **Time_t** *second_time)

Detailed Description

This library was created in order to provide some procedures and functions that are used to manage dates and times.

Author

Vincenzo Susso

Date

2019 Sep 09

Version

1.0 This library provide some procedures and functions to check if a date or time is valid, to set a valid value to a date or a time, and to compare different dates or times. This library was created following the standard ISO-8601, for more information visit: `https://en.wikipedia.org/wiki/ISO_8601` The time is shown is 24-hour format.

Macro Definition Documentation

#define CENTURY_LEAP_YEAR 400

This integer is used to check if a year century year is a leap year.

#define CENTURY_YEAR 100

This integer is used to check if a year is a leap year.

#define DATE_DELIMITER "/"

This string is used to separate the member of a date.

#define LEAP_YEAR 4

This integer is used to check if a year is a leap year.

#define MAX_DAY 31

This integer is used to indicates the maximum day that can be assigned to a valid date.

#define MAX_DAY_FEBRUARY 29

This integer is used to indicates the maximum day that can be assigned to a valid date in February.

#define MAX_HOUR 23

This integer is used to indicates the maximum hour of a valid time.

#define MAX_LENGHT_DATE_STRING_INPUT 11

This integer is used in order to indicates the maximum lenght of the string that will be used to take the date in input.

#define MAX_LENGHT_TIME_STRING_INPUT 6

This integer is used in order to indicates the maximum lenght of the string that will be used to take the time in input.

#define MAX_MINUTE 59

This integer is used to indicates the maximum minute of a valid time.

#define MIN_DAY 1

This integer is used to indicates the minimum day that can be assigned to a valid date.

#define MIN_HOUR 0

This integer is used to indicates the minimum hour of a valid time.

#define MIN_MINUTE 0

This integer is used to indicates the minimum minute of a valid time.

#define TIME_DELIMITER ":"

This string is used to separate the member of a time.

Enumeration Type Documentation

enum Date_order_t

This user-defined type is used in order to return a value after a date/time comparision.

Enumerator:

XE "olderDate.h"XE "Date.holder"older	The first date/time is older than the second one
XE "equalDate.h"XE "Date.hequal"equal	The first date/time and the second one are equal
XE "laterDate.h"XE "Date.hlater"later	The first date/time is later than the second one

enum Month_t

This user-defined type is used in order to indicates the months and improve the readability.

Enumerator:

XE "januaryDate.h"XE "Date.hjanuary"january	This member is used to indicate the month of January
XE "februaryDate.h"XE "Date.hfebruary"february	This member is used to indicate the month of February
XE "marchDate.h"XE "Date.hmarch"march	This member is used to indicate the month of March
XE "aprilDate.h"XE "Date.hapril"april	This member is used to indicate the month of April
XE "mayDate.h"XE "Date.hmay"may	This member is used to indicate the month of May
XE "juneDate.h"XE "Date.hjune"june	This member is used to indicate the month of June
XE "julyDate.h"XE "Date.hjuly"july	This member is used to indicate the month of July
XE "augustDate.h"XE "Date.haugust"august	This member is used to indicate the month of August
XE "septemberDate.h"XE "Date.hseptember"september	This member is used to indicate the month of September
XE "octoberDate.h"XE "Date.hoctober"october	This member is used to indicate the month of October
XE "novemberDate.h"XE	This member is used to indicate the month of November

"Date.hnovember"november	
XE "decemberDate.h"XE "Date.hdecember"december	This member is used to indicate the month of December

Function Documentation

Date_order_t cmpDate (const Date_t * *first_date*, const Date_t * *second_date*)

This functions is used to compare two dates.

Parameters

first_date	passed by pointer is used as a date to compare.
second_date	passed by pointer is used as a date to compare.

Returns

-1 if the first date is older than the second one, 0 if the first date is equal to the second one, 1 if the first date is later than the second one.

Date_order_t cmpTime (const Time_t * *first_time*, const Time_t * *second_time*)

Parameters

first_time	passed by pointer is used as a time to compare.
second_time	passed by pointer is used as a time to compare.

Returns

-1 if the first time is older than the second one, 0 if the first time is equal to the second one, 1 if the first time is later than the second one.

bool isLeapYear (const unsigned short *year*)

In the Gregorian calendar, every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400.

This function checks if the year is a leap year.

Parameters

year	is checked in order to see if it is a leap year.

Returns

true if the year is a leap year, otherwise It will return false.

bool isValidDate (const Date_t * *date*, const unsigned short *min_year*, const unsigned short *max_year*)

This function checks if the date that has been passed by pointer is a valid date and It is included between min_year and max_year.

Parameters

date	is checked in order to see if it is valid.
min_year	is the older valid year.
max_year	is the later valid year.

Returns

true if the date is a valid date, otherwise It will return false.

bool isValidTime (const Time_t * *time*)

This function checks if the time that has been passed by pointer is a valid time.

Parameters

time	is checked in order to see if it is valid.

Returns

true if the time is a valid time, otherwise It will return false.

void resetDate (Date_t * *date*)

This procedure reset the date passed by pointer assigning invalid values to the date.

Parameters

date	is resetted by the procedure.

void resetTime (Time_t * *time*)

This procedure reset the time passed by pointer assigning invalid values to the time.

Parameters

time	is resetted by the procedure

void setDate (Date_t * *date*, const unsigned short *min_year*, const unsigned short *max_year*, const char *printf_value*[])

This procedure sets a valid date to the date passed by pointer, the date must be included between min_year and max_year.

Parameters

date	is used to assigns a valid value.
min_year	is the older valid year.
max_year	is the later valid_year.
printf_value	says to the users what they should enter.

void setTime (Time_t * *time*, const char *printf_value*[])

This procedure sets a valid time to the time passed by pointer.

Parameters

time	is used to assigns a valid value.
printf_value	says to the users what they should enter.

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/File.c File Reference

This file is the implementation file of **File.h**.
`#include "File.h"`

Functions

- **File_status_t isValidFile** (const char path_file[])
- **File_status_t deleteFile** (const char path_file[])
- **File_status_t writeFile** (const char path_file[], void *pointer, size_t pointer_size, long int offset, int whence)
- **File_status_t readFile** (const char path_file[], void *pointer, size_t pointer_size, long int offset, int whence)
- long int **getLastIndex** (const char path_file[])
- int **getNumberRecord** (const char path_file[], size_t size_record)

Detailed Description

This file is the implementation file of **File.h**.

Author

Vincenzo Susso

Date

2019 Sep 10

Version

1.0

Function Documentation

deleteFile (const char *path_file*[])

This function will delete the file specified in the path_file.

Parameters

path_file	is the path of the file to delete.

Returns

2 if the file has been deleted, otherwise this function will return 0.

getLastIndex (const char *path_file*[])

This function will return the last index of the file specified by the path passed by pointer.

Parameters

path_file	is the path of the file that will be read to get the last index of the file

Returns

the index of the file if it has been found, otherwise It will return -1

getNumberRecord (const char *path_file*[], size_t *size_record*)

This function will return the number of records that have been saved into the file specified by the path passed by pointer.

Parameters

path_file	is the path of the file that will be read to get the number of records that have

	been saved into the file
size_record	is the size of records that have been saved into the file

Returns

the number of records that have been saved into the file, otherwise It will return 0

isValidFile (const char *path_file*[])

This function checks if the directory and the file specified in the path_file exist, otherwise this function will create the directory and the file.

Parameters

path_file	is the path of the file to check or create.

Returns

2 if the directory and the file exist or if they have been created, otherwise It will return 0.

readFile (const char *path_file*[], void * *pointer*, size_t *pointer_size*, long int *offset*, int *whence*)

This function will read the file specified by the path passed by pointer whence the offset is specified.

Parameters

path_file	is the path of the file to read
pointer	will point the element that will be read to the file
pointer_size	is the size of the pointer that will be read to the file
offset	is where the file will be read
whence	is where the file will start to count the offset

Returns

2 if the file has been read, 1 if the file has reached EOF, otherwise this function will return 0

writeFile (const char *path_file*[], void * *pointer*, size_t *pointer_size*, long int *offset*, int *whence*)

This function will write the file specified by the path passed by pointer whence the offset is specified.

Parameters

path_file	is the path of the file to write
pointer	will point the element that will be written to the file
pointer_size	is the size of the pointer that will be written to the file
offset	is where the file will be written
whence	is where the file will start to count the offset

Returns

2 if the file has been written, otherwise this function will return 0

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/File.h File Reference

This library was created in order to provide some procedures and functions that are used to files.

```
#include <stdbool.h>
#include <stdio.h>
#include <errno.h>
#include <direct.h>
```

Macros

- #define **DIRECTORY_PATH** "../Files"
- #define **NUMBER_MEMBER_FILE** 1
- #define **INDEX_NOT_FOUND** -1

Enumerations

- enum **File_status_t** { **error_file** = 0, **fail**, **done** }

Functions

- **File_status_t isValidFile** (const char path_file[])
- **File_status_t deleteFile** (const char path_file[])
- **File_status_t writeFile** (const char path_file[], void *pointer, size_t pointer_size, long int offset, int whence)
- **File_status_t readFile** (const char path_file[], void *pointer, size_t pointer_size, long int offset, int whence)
- long int **getLastIndex** (const char path_file[])
- int **getNumberRecord** (const char path_file[], size_t size_record)

Detailed Description

This library was created in order to provide some procedures and functions that are used to files.

Author

Vincenzo Susso

Date

2019 Sep 10

Version

1.0 This library can check if a file exits and can create new files, furthermore this library can read and write a file. This library can get the number of records that have been saved into a file.

Macro Definition Documentation

#define DIRECTORY_PATH "../Files"

This string is used to indicates the relative path of the directory that will store all the files.

#define INDEX_NOT_FOUND -1

This integer is used to indicates that the index of a record has not been found.

#define NUMBER_MEMBER_FILE 1

This integer indicates the number of member that the file can read/write. This integer can be used to check if the file has been read or written correctly.

Enumeration Type Documentation

enum File_status_t

This user-defined type is used to manage the operation with files.

Enumerator:

XE "error_fileFile.h"XE "File.herror_file"error_file	This value is returned when a fatal error has occurred
XE "failFile.h"XE "File.hfail"fail	This value is returned when a minor error has occurred so the program can continue to run
XE "doneFile.h"XE "File.hdone"done	This value is returned when error has not occurred

---

Function Documentation

File_status_t deleteFile (const char *path_file*[])

This function will delete the file specified in the path_file.

Parameters

path_file	is the path of the file to delete.

Returns

2 if the file has been deleted, otherwise this function will return 0.

long int getLastIndex (const char *path_file*[])

This function will return the last index of the file specified by the path passed by pointer.

Parameters

path_file	is the path of the file that will be read to get the last index of the file

Returns

the index of the file if it has been found, otherwise It will return -1

int getNumberRecord (const char *path_file*[], size_t *size_record*)

This function will return the number of records that have been saved into the file specified by the path passed by pointer.

Parameters

path_file	is the path of the file that will be read to get the number of records that have been saved into the file
size_record	is the size of records that have been saved into the file

Returns

the number of records that have been saved into the file, otherwise It will return 0

File_status_t isValidFile (const char *path_file*[])

This function checks if the directory and the file specified in the path_file exist, otherwise this function will create the directory and the file.

Parameters

path_file	is the path of the file to check or create.

Returns

2 if the directory and the file exist or if they have been created, otherwise It will return 0.

File_status_t readFile (const char *path_file*[], void * *pointer*, size_t *pointer_size*, long int *offset*, int *whence*)

This function will read the file specified by the path passed by pointer whence the offset is specified.

Parameters

path_file	is the path of the file to read
pointer	will point the element that will be read to the file
pointer_size	is the size of the pointer that will be read to the file
offset	is where the file will be read
whence	is where the file will start to count the offset

Returns

2 if the file has been read, 1 if the file has reached EOF, otherwise this function will return 0

File_status_t writeFile (const char *path_file*[], void * *pointer*, size_t *pointer_size*, long int *offset*, int *whence*)

This function will write the file specified by the path passed by pointer whence the offset is specified.

Parameters

path_file	is the path of the file to write
pointer	will point the element that will be written to the file
pointer_size	is the size of the pointer that will be written to the file
offset	is where the file will be written
whence	is where the file will start to count the offset

Returns

2 if the file has been written, otherwise this function will return 0

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/main.c File Reference

main file

```
#include "main.h"
```

Functions

- int **main** (void)
- void **introduction** (void)
- void **showMenu** (void)

Detailed Description

main file

Author

Vincenzo Susso

Date

2019 Sep 10

Version

1.0

Function Documentation

introduction (void)

This procedure is used in order to show an introduction during the starting of the program.

int main (void)

showMenu (void)

This procedure is used to show the menu to the user.

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/main.h File Reference

This library was developed in order to manage menu and files.

```
#include <stdio.h>
#include <stdlib.h>
#include "CUnit/Basic.h"
#include "Carpool.h"
```

Macros

- #define **DRIVERS_FILE_PATH** "../Files/Drivers.dat"
- #define **TRAVELS_FILE_PATH** "../Files/Travels.dat"
- #define **RATINGS_FILE_PATH** "../Files/Ratings.dat"
- #define **ID_FILE_PATH** "../Files/ID.dat"

Enumerations

- enum **Menu_choice_t** { **add_driver**, **edit_driver**, **delete_driver**, **show_all_drivers**, **add_travel**, **edit_travel**, **delete_travel**, **show_all_travels**, **book_travel**, **evaluate_driver**, **sort_drivers**, **sort_travels**, **exit_menu**, **not_valid_choice** }

Functions

- void **introduction** (void)
- void **showMenu** (void)

Detailed Description

This library was developed in order to manage menu and files.

Author

Vincenzo Susso

Date

2019 Sep 10

Version

1.0 This library is used to define files that are used to store drivers, travels, ratings and IDs. This library also provide procedure to show menu and an introduction during the starting of the program.

Macro Definition Documentation

#define DRIVERS_FILE_PATH "../Files/Drivers.dat"

This string indicates the relative path of the file that will store the drivers.

This string indicates the relative path of the file that will store the IDs for drivers and travels.

#define ID_FILE_PATH "../Files/ID.dat"

#define RATINGS_FILE_PATH "../Files/Ratings.dat"

This string indicates the relative path of the file that will store the ratings.

#define TRAVELS_FILE_PATH "../Files/Travels.dat"

This string indicates the relative path of the file that will store the travels.

Enumeration Type Documentation

enum Menu_choice_t

This user-defined type is used in order to define the option of the menu, this user-defined type was also created in order to improve the readability.

Enumerator:

XE "add_drivermain.h"XE "main.hadd_driver"add_driver	This member allow the user to add a driver to the system
XE "edit_drivermain.h"XE "main.hedit_driver"edit_driver	This member allow the user to edit a member of a driver
XE "delete_drivermain.h"XE "main.hdelete_driver"delete_driver	This member allow the user to delete a driver to the system
XE "show_all_driversmain.h"XE "main.hshow_all_drivers"show_all_drivers	This member allow the user to show all the drivers
XE "add_travelmain.h"XE "main.hadd_travel"add_travel	This member allow the user to add a travel to the system
XE "edit_travelmain.h"XE "main.hedit_travel"edit_travel	This member allow the user to edit a member of a travel
XE "delete_travelmain.h"XE "main.hdelete_travel"delete_travel	This member allow the user to delete a travel to the system
XE "show_all_travelsmain.h"XE "main.hshow_all_travels"show_all_travels	This member allow the user to show all the travels
XE "book_travelmain.h"XE "main.hbook_travel"book_travel	This member allow the user to book a travel
XE "evaluate_drivermain.h" XE "main.hevaluate_driver" evaluate_driver	This member allow the user to evaluate a the driver
XE	This member allow the user to sort all the drivers

"sort_driversmain.h"XE "main.hsort_drivers"sort_drivers	
XE "sort_travelsmain.h"XE "main.hsort_travels"sort_travels	This member allow the user to sort all the travels
XE "exit_menumain.h"XE "main.hexit_menu"exit_menu	This member allow the user to exit from the program
XE "not_valid_choicemain.h"XE "main.hnot_valid_choice"not_valid_choice	This member is used to set a not valid choice to the menu option

Function Documentation

void introduction (void)

This procedure is used in order to show an introduction during the starting of the program.

void showMenu (void)

This procedure is used to show the menu to the user.

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/Utilities.c File Reference

This file is the implementation file of **Utilities.h**.
```
#include "Utilities.h"
```

Functions

- void **clearBuffer** (void)
- void **initializeCMD** (void)
- void **printfError** (const char string[])
- void **addNullCharacterString** (char string[])
- void **capitalizeString** (char string[])
- bool **isIncluded** (const int min, const int max, const int number)
- bool **isLatinString** (const char string[])
- bool **isNumberString** (const char string[])
- bool **isVoidString** (const char string[])
- bool **isEmail** (const char **email**[])
- bool **isPassword** (const char **password**[])
- bool **isPhoneNumber** (const char **phone_number**[])
- bool **isDecimalNumber** (const char decimal_number[])
- bool **cmpString** (char first_string[], char second_string[])
- double **getSecondSort** (const time_t start, const time_t end)

Detailed Description

This file is the implementation file of **Utilities.h**.

Author

Vincenzo Susso

Date

2019 Sep 09

Version

1.0

Function Documentation

addNullCharacterString (char *string*[])

The procedure adds the null character to the string in order to indicates the end of the string.

Parameters

string	null character is added.

capitalizeString (char *string*[])

This procedure converts the first letter of the string to uppercase and the other ones to lowercase.

Parameters

string	is converted in a string with the first character in uppercase and the other ones to lowercase.

clearBuffer (void)

This procedure is used to clear the buffer after an input, this procedure delete the '\n' character that usually remains into the buffer after a scanf().

cmpString (char *first_string*[], char *second_string*[])

This function is used to compare two strings. If the matching of the strings is equal or above the 70%, the string will return true.

Parameters

first_string	is the first member to compare.
second_string	is the second member to compare.

Returns

true if the matching of the strings is equal or abouve the 70%, otherwise this funtion will return false.

getSecondSort (const time_t *start*, const time_t *end*)

This function is used to calculate the number of second that the sort has spent.

Parameters

start	indicates when the sorting starts.
end	indicates when the sorting ends.

Returns

the seconds that the sorting algorithms has spent.

initializeCMD (void)

This procedure is used to enable ANSI escape sequences on CMD.

isDecimalNumber (const char *decimal_number*[])

This function checks if the decimal number is valid.

Parameters

decimal_number	is checked in order to see if It is valid.

Returns

true if the decimal number is valid, otherwise It will return false.

isEmail (const char *email*[])

The format of email addresses is "local-part@domain". The local-part of the email address may use any of these ASCII characters:

- Uppercase and lowercase Latin letters A to Z and a to z;
- Digits 0 to 9;
- Dot ".", provided that it is not the first or last character, and provided also that it does not appear consecutively;

The domain name part of an email address has to conform to strict guidelines:

- Uppercase and lowercase Latin letters A to Z and a to z;
- Digits 0 to 9, provided that top-level domain names are not all-numeric; For more informations, please visit here: `https://en.wikipedia.org/wiki/Email_address`

This function checks if email is valid. The email should be of the format "localname@domain".

Parameters

email	is checked in order to see if It is valid.

Returns

true if the email is valid, otherwise It will return false.

isIncluded (const int *min*, const int *max*, const int *number*)

This function checks if the number is included between min and max.

Parameters

min	is the minimum valid value.
max	is the maximum valid value.
number	is what the user wants to test.

Returns

true if number is included between min and max, otherwise It will return false.

isLatinString (const char *string*[])

This function checks if each character of the string belongs to the Latin alphabet.

Parameters

string	is checked in order to see if each character of the string belongs to the Latin alphabet.

Returns

true if each character of the string belongs to the Lating alphabet, otherwise It will return false.

isNumberString (const char *string*[])

This function checks if each character of the string is a digit.

Parameters

string	is checked in order to see if each character of the string is a digit.

Returns

true if each character of the string is a digit, otherwise It will return false.

isPassword (const char *password*[])

This function checks if the password is valid. The password should have at least one character uppercase and one digit.

Parameters

password	is checked in order to see if It is valid.

Returns

true if the password is valid, otherwise It will return false.

isPhoneNumber (const char *phone_number*[])

This function checks if the phone number is valid. The phone number should be of the format "+xxx xxxxxxxxxxxx"

Parameters

phone_number	is checked in order to see if It is valid.

Returns

true if the phone number is valid, otherwise It will return false.

isVoidString (const char *string*[])

This function checks if the string is void.

Parameters

string	is checked in order to see if the string is void.

Returns

true if the string is void, otherwise It will return false.

printfError (const char *string*[])

This procedure printf the string in red.

Parameters

string	is printed in red.

C:/Users/WinEnzo/Documents/Eclipse/Caso_di_Studio-Carpool/src/Utilities.h File Reference

This library was created in order to provide some utility procedures and functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <ctype.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <windows.h>
```

Macros

- #define **NEWLINE_CHARACTER** '\n'
- #define **NEWLINE_STRING** "\n"
- #define **SPACE_STRING** " "
- #define **NULL_STRING** "\0"
- #define **PERIOD_CHARACTER** '.'
- #define **AT_SIGN_STRING** "@"
- #define **NUMBER_DOT_DOMAIN** 1
- #define **PLUS_CHARACTER** '+'
- #define **MAX_LENGHT_COUNTRY_CODE** 4
- #define **MAX_LENGHT_SUBSCRIBER_NUMBER** 12
- #define **MIN_UPPERCASE_CHARACTERS** 1
- #define **MIN_NUMBER_CHARACTERS** 1
- #define **SPACE_CHARACTER** ' '
- #define **BASE_STRTOL** 10
- #define **PERIOD_STRING** "."
- #define **MATCHING_PERCENT** 70
- #define **FLOOR_ROUNDING** 0.5
- #define **ENABLE_VIRTUAL_TERMINAL_PROCESSING** 0x0004
- #define **RESET** "\033[0m"
- #define **RED** "\033[31m"

Functions

- void **clearBuffer** (void)
- void **initializeCMD** (void)
- void **printfError** (const char string[])
- void **addNullCharacterString** (char string[])
- void **capitalizeString** (char string[])
- bool **isIncluded** (const int min, const int max, const int number)
- bool **isNumberString** (const char string[])
- bool **isLatinString** (const char string[])
- bool **isVoidString** (const char string[])
- bool **isEmail** (const char **email**[])
- bool **isPassword** (const char **password**[])
- bool **isPhoneNumber** (const char **phone_number**[])
- bool **isDecimalNumber** (const char decimal_number[])
- bool **cmpString** (char first_string[], char second_string[])
- double **getSecondSort** (const time_t start, const time_t end)

Detailed Description

This library was created in order to provide some utility procedures and functions.

Author

Vincenzo Susso

Date

2019 Sep 09

Version

1.0 This library provide several utility, for example, the procedure cleanBuffer() can clear the buffer after an input, some function are used in order to check the correct insertion of a string and some function are used to operate on strings.

Macro Definition Documentation

#define AT_SIGN_STRING "@"

This string is used in order to check if the email has an at sign character "@".

#define BASE_STRTOL 10

This integer is used as base during the convertion of a string to long.

#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004

This hexadecimal is used in order to enable ANSI escape sequences on CMD (pre Windows 10).

#define FLOOR_ROUNDING 0.5

This float is used to approximate a decimal number correctly. The function floor(double x) return the number x rounded dows, so, because of this i need this const. E.g: floor(3.1) = 3 floor(3.8) = 3 floor(4.2) = 4 Using const: floor(3.1 + FLOOR_ROUNDING) = 3 floor(3.8 + FLOOR_ROUNDING) = 4

#define MATCHING_PERCENT 70

This integer is used when comparing two strings to indicate the percentage needed.

#define MAX_LENGHT_COUNTRY_CODE 4

This integer is used to check the correct number of characters that made the country code, this integer is obtained by: Plus character "+" + Country code lenght (3).

#define MAX_LENGHT_SUBSCRIBER_NUMBER 12

This integer is used to check the correct number of characters that made the subscriber number.

#define MIN_NUMBER_CHARACTERS 1

This integer is used to check if the password has the minimum number of digits.

#define MIN_UPPERCASE_CHARACTERS 1

This integer is used to check if the password has the minimum number of uppercase characters.

#define NEWLINE_CHARACTER '\n'

This character is used in order to clean the buffer after an input.

#define NEWLINE_STRING "\n"

This character is used in order to check if the input string is a void string.

#define NULL_STRING "\0"

This character is used in order to check if the input string is a void string.

#define NUMBER_DOT_DOMAIN 1

This integer is used in order to check if the domain of the email has the right number of dots.

#define PERIOD_CHARACTER '.'

This character is used in order to check if the email has a valid domain.

#define PERIOD_STRING "."

This string is used to check if a decimal number has a period.

#define PLUS_CHARACTER '+'

This character is used in order to check if the number has a valid country code.

#define RED "\033[31m"

This escape sequence makes the text red.

#define RESET "\033[0m"

This escape sequence resets the color of the text.

#define SPACE_CHARACTER ' '

This character is used to check if the password doesn't have a space.

#define SPACE_STRING " "

This character is used in order to check if the input string is a void string.

Function Documentation

void addNullCharacterString (char *string*[])

The procedure adds the null character to the string in order to indicates the end of the string.

Parameters

string	null character is added.

void capitalizeString (char *string*[])

This procedure converts the first letter of the string to uppercase and the other ones to lowercase.

Parameters

string	is converted in a string with the first character in uppercase and the other ones to lowercase.

void clearBuffer (void)

This procedure is used to clear the buffer after an input, this procedure delete the '\n' character that usually remains into the buffer after a scanf().

bool cmpString (char *first_string*[], char *second_string*[])

This function is used to compare two strings. If the matching of the strings is equal or above the 70%, the string will return true.

Parameters

first_string	is the first member to compare.
second_string	is the second member to compare.

Returns

true if the matching of the strings is equal or abouve the 70%, otherwise this funtion will return false.

double getSecondSort (const time_t *start*, const time_t *end*)

This function is used to calculate the number of second that the sort has spent.

Parameters

start	indicates when the sorting starts.
end	indicates when the sorting ends.

Returns

the seconds that the sorting algorithms has spent.

void initializeCMD (void)

This procedure is used to enable ANSI escape sequences on CMD.

bool isDecimalNumber (const char *decimal_number*[])

This function checks if the decimal number is valid.

Parameters

decimal_number	is checked in order to see if It is valid.

Returns

true if the decimal number is valid, otherwise It will return false.

bool isEmail (const char *email*[])

The format of email addresses is "local-part@domain". The local-part of the email address may use any of these ASCII characters:

- Uppercase and lowercase Latin letters A to Z and a to z;
- Digits 0 to 9;
- Dot ".", provided that it is not the first or last character, and provided also that it does not appear consecutively;

The domain name part of an email address has to conform to strict guidelines:

- Uppercase and lowercase Latin letters A to Z and a to z;
- Digits 0 to 9, provided that top-level domain names are not all-numeric; For more informations, please visit here: `https://en.wikipedia.org/wiki/Email_address`

This function checks if email is valid. The email should be of the format "localname@domain".

Parameters

email	is checked in order to see if It is valid.

Returns

true if the email is valid, otherwise It will return false.

bool isIncluded (const int *min*, const int *max*, const int *number*)

This function checks if the number is included between min and max.

Parameters

min	is the minimum valid value.

max	is the maximum valid value.
number	is what the user wants to test.

Returns

true if number is included between min and max, otherwise It will return false.

bool isLatinString (const char *string*[])

This function checks if each character of the string belongs to the Latin alphabet.

Parameters

string	is checked in order to see if each character of the string belongs to the Latin alphabet.

Returns

true if each character of the string belongs to the Lating alphabet, otherwise It will return false.

bool isNumberString (const char *string*[])

This function checks if each character of the string is a digit.

Parameters

string	is checked in order to see if each character of the string is a digit.

Returns

true if each character of the string is a digit, otherwise It will return false.

bool isPassword (const char *password*[])

This function checks if the password is valid. The password should have at least one character uppercase and one digit.

Parameters

password	is checked in order to see if It is valid.

Returns

true if the password is valid, otherwise It will return false.

bool isPhoneNumber (const char *phone_number*[])

This function checks if the phone number is valid. The phone number should be of the format "+xxx xxxxxxxxxxxx"

Parameters

phone_number	is checked in order to see if It is valid.

Returns

true if the phone number is valid, otherwise It will return false.

bool isVoidString (const char *string*[])

This function checks if the string is void.

Parameters

string	is checked in order to see if the string is void.

Returns

true if the string is void, otherwise It will return false.

void printfError (const char *string*[])

This procedure printf the string in red.

Parameters

string	is printed in red.