Meteologica

Generado por Doxygen 1.8.13

Índice general

Capítulo 1

Índice de clases

1.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 3

Documentación de las clases

3.1. Referencia de la Estructura City

Estructura de datos que contiene la informacíon de la ciudad almacenada.

`#include <meteo.h>`

Atributos públicos

- char city [100]
- int n_meteoData_entries
- int memory_reserved
- struct Meteo * meteoData_array

3.1.1. Descripción detallada

Estructura de datos que contiene la informacíon de la ciudad almacenada.

3.1.2. Documentación de los datos miembro

3.1.2.1. city

`char City::city[100]`

Nombre de la ciudad

3.1.2.2. memory_reserved

`int City::memory_reserved`

Número de espacio reservado para entradas meteorológicas

3.1.2.3. meteoData_array

```
struct Meteo* City::meteoData_array
```

Array con la infomación meteorológica para la ciudad

3.1.2.4. n_meteoData_entries

```
int City::n_meteoData_entries
```

Número de entradas meteorológicas almacenadas para esta ciudad

La documentación para esta estructura fue generada a partir del siguiente fichero:

- meteo.h

3.2. Referencia de la Estructura Meteo

Estructura de datos que continen la información meteorilógica de una ciudad en una fecha determinada.

```
#include <meteo.h>
```

Atributos públicos

- char date [50]
- char city [100]
- char max_temp [50]
- char min_temp [50]
- char precipitations [50]
- char cloudiness [50]

3.2.1. Descripción detallada

Estructura de datos que continen la información meteorilógica de una ciudad en una fecha determinada.

3.2.2. Documentación de los datos miembro

3.2.2.1. city

```
char Meteo::city[100]
```

Nombre de la ciudad de la entrada meteorológica

3.2.2.2. cloudiness

```
char Meteo::cloudiness[50]
```

Nubosidad de lea entrada meteorológica

3.2.2.3. date

```
char Meteo::date[50]
```

Fecha de la entrada meteorológica

3.2.2.4. max_temp

```
char Meteo::max_temp[50]
```

Temperatura máxima de la entrada meteorológica

3.2.2.5. min_temp

```
char Meteo::min_temp[50]
```

Temperatura mínima de la entrada meteorológica

3.2.2.6. precipitations

```
char Meteo::precipitations[50]
```

Precipitaciones de la entrada meteorológica

La documentación para esta estructura fue generada a partir del siguiente fichero:

- meteo.h

Capítulo 4

Documentación de archivos

4.1. Referencia del Archivo helpers.c

En este fichero están definidas las funciones declaradas en helpers.h.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <limits.h>
#include "helpers.h"
```

Funciones

- char ∗∗ split (char ∗string, const char delimiter)

  La función split() divide el string recibido en substring utilizando como divisor un delimitador. Ejemplo split("Esto es una prueba", ' ') obtendriamos un array con 4 strings "Esto", "es", "una", "prueba".
- void fixDecimalNotation (char ∗string)

  La función fixDecimalNotation() corrige la notación de los números decimales. Sustituye los decimales expresados con ',' por '.' Ejemplo: 12,23 -> 12.23.
- void trim (char ∗string)

  La función trim() elimina los espacios en blanco ' ' y otros caracters especiales tanto al comienzo como al final del string recibido.
- int getIntFromUserInput (int ∗number)

  La función getIntFromUserInput() recibe una entrada por teclado y la convierte en INT cuando sea posible.
- int getStringFromUserInput (char ∗string, int size)

  La función getStringFromUserInput() recibe una entrada por teclado y la almacena.
- int compareString (void const ∗str1, void const ∗str2)

  La función compareString() compara dos string. Internamente utiliza la funcion strcmp(). La razón de ser de esta función es para utilizarla como comparador en la función de ordenación qsort()
- int compareDates (char ∗date1, char ∗date2)

  La función compareDates() compara dos fechas en formato AAAA/MM/DD (Año/Mes/Día)
- int strCelsiusToStrFahrenheit (char ∗celsius, char ∗∗fahrenheit)

  La función strCelsiusToStrFahrengeit() recibe grados celsiud en formato string para transofrmalor en grados Fahrenheit también en formato string.
- int stringToFloat (double ∗value, char ∗data)

La función stringToFloat() convierte una cadena de caracteres en float si es posible.

- double celsiusToFahrenheit (double celsius)

  La función celsiusToFahrenheit convierte los grados Celius introducidos en Fahrenheit.

- int storeToFile (char *filename, char *data)

  La función storeToFile() almacena en un fichero los datos proporcionados. Esta función creará el fichero si no existe o lo sobreescribirá en caso de que existiera.

- void replace (char *string, char c1, char c2)

  La función replace(), reemplaza la el caracter c1 por c2 en toda la string.

4.1.1. Descripción detallada

En este fichero están definidas las funciones declaradas en helpers.h.

4.1.2. Documentación de las funciones

4.1.2.1. celsiusToFahrenheit()

```
double celsiusToFahrenheit (
            double celsius )
```

La función celsiusToFahrenheit convierte los grados Celius introducidos en Fahrenheit.

Parámetros

celsius	Se trata de un double que contiene los grados celsius a convertir

Devuelve

La función celsiusToFahrenheit devuelve un double con la conversión de Celsius a Fahrenheit

4.1.2.2. compareDates()

```
int compareDates (
            char * date1,
            char * date2 )
```

La función compareDates() compara dos fechas en formato AAAA/MM/DD (Año/Mes/Día)

Parámetros

date1	Se trata de un string (char *) que contiene la primera fecha a comparar.
date2	Se trata de un string (char *) que contiene la segunda fecha a comparar.

Devuelve

Si date1 > date2 devuelve un 1; Si date1 < date2 devuelve un -1; Si date1 == date2 devuelve un 0;

4.1.2.3. compareString()

```
int compareString (
            void const * str1,
            void const * str2 )
```

La función compareString() compara dos string. Internamente utiliza la funcion strcmp(). La razón de ser de esta función es para utilizarla como comparador en la función de ordenación qsort()

Parámetros

str1	Se trata de un string (char *) que contiene la primera string a comparar.
str2	Se trata de un string (char *) que contiene la segunda string a comparar.

Devuelve

Si str1 > str2 devuelve un número positivo; Si str1 < str2 devuelve un número negativo; Si str1 == str2 devuelve un cero;

4.1.2.4. fixDecimalNotation()

```
void fixDecimalNotation (
            char * string )
```

La función fixDecimalNotation() corrige la notación de los números decimales. Sustituye los decimales expresados con ',' por '.' Ejemplo: 12,23 -> 12.23.

Parámetros

string	Se trata de un string (char *) que contiene el número al que se le quiere corregir el simbolo de decimal.

4.1.2.5. getIntFromUserInput()

```
int getIntFromUserInput (
            int * number )
```

La función getIntFromUserInput() recibe una entrada por teclado y la convierte en INT cuando sea posible.

Parámetros

number	Se trata de un puntero a entero (int *) en el cuál se almacenará el número recibido.

Devuelve

La función devuelve un 0 en caso de que se haya podido recibir y convertir la entrada a entero. En caso de error, devolverá -1.

4.1.2.6. getStringFromUserInput()

```
int getStringFromUserInput (
            char * string,
            int size )
```

La función getStringFromUserInput() recibe una entrada por teclado y la almacena.

Parámetros

string	Se trata de un string (char *) en el cuál se almacenará el la cadena de texto recibida.
size	Entero con la tamaño máximo de la cadena que se desea recibir.

Devuelve

La función devuelve un entero indicando la longitud del string recibido. En caso de error devolverá -1.

4.1.2.7. replace()

```
int replace (
            char * string,
            char c1,
            char c2 )
```

La función replace(), reemplaza la el caracter c1 por c2 en toda la string.

Parámetros

string	Se trata de un string que contiene los caracteres que se quieren reemplazar
c1	Se trata de caracter que se quiere reemplazar por otro
c2	Se trata de caracter por el cuál se va a reemplazar

4.1.2.8. split()

```
char ** split (
            char * string,
            const char delimiter )
```

La función split() divide el string recibido en substring utilizando como divisor un delimitador. Ejemplo split("Esto es una prueba", ' ') obtendriamos un array con 4 strings "Esto", "es", "una", "prueba".

Parámetros

string	Se trata de un string (char $*$) que contiene la string que se quiere dividir.
delimiter	Se trata de un caracter (char) que contiene el delimitador por el cual se dividirá el string.

Devuelve

La función split() devuelve un array de strings (char $**$) en el cual cada posición del array contiene una de las divisiones del string introducido.

Atención

Es necesario liberar la memoria reservada por la funcion split() del parametro devuelto. Para ello liberaremos la memoria reservada para cada string devuelto en el array y luego liberaremos la memoria del array.

4.1.2.9. storeToFile()

```
int storeToFile (
            char * filename,
            char * data )
```

La función storeToFile() almacena en un fichero los datos proporcionados. Esta función creará el fichero si no existe o lo sobreescribirá en caso de que existiera.

Parámetros

filename	Se trata de un string (char $*$) que contiene el nombre del fichero donde se va a guardar los datos
data	Se trata de un string (char $*$) que contiene los datos que se van a almancenar en el fichero

Devuelve

La función storeToFile devuelve un entero con un valor de 0 en caso de éxito y un -1 en caso de error

4.1.2.10. strCelsiusToStrFahrenheit()

```
int strCelsiusToStrFahrenheit (
            char * Celsius,
            char ** Fahrenheit )
```

La función strCelsiusToStrFahrengeit() recibe grados celsiud en formato string para transofrmalor en grados Fahrenheit también en formato string.

Parámetros

celsius	Se trata de un string (char $*$) que contiene los grados celsiud a convertir
fahrenheit	Se trata de un puntero a string (char $**$) que contendrá el resultado de la conversión de Celsiud a Fahrenheit

Devuelve

La función strCelsiudToStrFahrenheit() devuelve un entero con valor 0 si la operación a resultado con éxito y un -1 si se produce un error.

Atención

Es necesario liberar la memoria reservada por la función strCelsiusToStrFahrenheit(). Para ello liberaremos la mermoria del puntero a string fahrenheit.

4.1.2.11. stringToFloat()

```
int stringToFloat (
            double * value,
            char * data )
```

La función stringToFloat() convierte una cadena de caracteres en float si es posible.

Parámetros

value	Se trata de un double donde se almacenará el número en caso de éxito;
data	Se trata de un string (char $*$) que contiene los datos que se desean convertir

Devuelve

La función stringToFloat devuelve un entero con un 0 en caso de éxito y un -1 si hay algún error.

4.1.2.12. trim()

```
void trim (
            char * string )
```

La función trim() elimina los espacios en blanco ' ' y otros caracters especiales tanto al comienzo como al final del string recibido.

Parámetros

string	Se trata de un string (char ∗) a la que le queremos quitar los espacios en blanco.

4.2. Referencia del Archivo helpers.h

En este fichero están declaradas aquellas funciones cuyo objetivo es de caracter general y pueden ser invocadas en multiples situaciones.

Funciones

- char ∗∗ split (char ∗string, const char delimiter)

  La función split() divide el string recibido en substring utilizando como divisor un delimitador. Ejemplo split("Esto es una prueba", ' ') obtendriamos un array con 4 strings "Esto", "es", "una", "prueba".
- void fixDecimalNotation (char ∗string)

  La función fixDecimalNotation() corrige la notación de los números decimales. Sustituye los decimales expresados con ',' por '.' Ejemplo: 12,23 -> 12.23.
- void trim (char ∗string)

  La función trim() elimina los espacios en blanco ' ' y otros caracters especiales tanto al comienzo como al final del string recibido.
- int getIntFromUserInput (int ∗number)

  La función getIntFromUserInput() recibe una entrada por teclado y la convierte en INT cuando sea posible.
- int getStringFromUserInput (char ∗string, int size)

  La función getStringFromUserInput() recibe una entrada por teclado y la almacena.
- int compareString (void const ∗str1, void const ∗str2)

  La función compareString() compara dos string. Internamente utiliza la funcion strcmp(). La razón de ser de esta función es para utilizarla como comparador en la función de ordenación qsort()
- int compareDates (char ∗date1, char ∗date2)

  La función compareDates() compara dos fechas en formato AAAA/MM/DD (Año/Mes/Día)
- int strCelsiusToStrFahrenheit (char ∗celsius, char ∗∗fahrenheit)

  La función strCelsiusToStrFahrengeit() recibe grados celsiud en formato string para transofrmalor en grados Fahrenheit también en formato string.
- int stringToFloat (double ∗value, char ∗data)

  La función stringToFloat() convierte una cadena de caracteres en float si es posible.
- double celsiusToFahrenheit (double celsius)

  La función celsiusToFahrenheit convierte los grados Celius introducidos en Fahrenheit.
- int storeToFile (char ∗filename, char ∗data)

  La función storeToFile() almacena en un fichero los datos proporcionados. Esta función creará el fichero si no existe o lo sobreescribirá en caso de que existiera.
- void replace (char ∗string, char c1, char c2)

  La función replace(), reemplaza la el caracter c1 por c2 en toda la string.

4.2.1. Descripción detallada

En este fichero están declaradas aquellas funciones cuyo objetivo es de caracter general y pueden ser invocadas en multiples situaciones.

4.2.2. Documentación de las funciones

4.2.2.1. celsiusToFahrenheit()

```
double celsiusToFahrenheit (
            double celsius )
```

La función celsiusToFahrenheit convierte los grados Celius introducidos en Fahrenheit.

Parámetros

celsius	Se trata de un double que contiene los grados celsius a convertir

Devuelve

La función celsiusToFahrenheit devuelve un double con la conversión de Celsius a Fahrenheit

4.2.2.2. compareDates()

```
int compareDates (
            char * date1,
            char * date2 )
```

La función compareDates() compara dos fechas en formato AAAA/MM/DD (Año/Mes/Día)

Parámetros

date1	Se trata de un string (char $*$) que contiene la primera fecha a comparar.
date2	Se trata de un string (char $*$) que contiene la segunda fecha a comparar.

Devuelve

Si date1 $>$ date2 devuelve un 1; Si date1 $<$ date2 devuelve un -1; Si date1 == date2 devuelve un 0;

4.2.2.3. compareString()

```
int compareString (
            void const * str1,
            void const * str2 )
```

La función compareString() compara dos string. Internamente utiliza la funcion strcmp(). La razón de ser de esta función es para utilizarla como comparador en la función de ordenación qsort()

Parámetros

str1	Se trata de un string (char $*$) que contiene la primera string a comparar.
str2	Se trata de un string (char $*$) que contiene la segunda string a comparar.

Devuelve

Si str1 $>$ str2 devuelve un número positivo; Si str1 $<$ str2 devuelve un número negativo; Si str1 == str2 devuelve un cero;

4.2.2.4. fixDecimalNotation()

```
void fixDecimalNotation (
            char * string )
```

La función fixDecimalNotation() corrige la notación de los números decimales. Sustituye los decimales expresados con ',' por '.' Ejemplo: 12,23 -$>$ 12.23.

Parámetros

string	Se trata de un string (char $*$) que contiene el número al que se le quiere corregir el simbolo de decimal.

4.2.2.5. getIntFromUserInput()

```
int getIntFromUserInput (
            int * number )
```

La función getIntFromUserInput() recibe una entrada por teclado y la convierte en INT cuando sea posible.

Parámetros

number	Se trata de un puntero a entero (int $*$) en el cuál se almacenará el número recibido.

Devuelve

La función devuelve un 0 en caso de que se haya podido recibir y convertir la entrada a entero. En caso de error, devolverá -1.

4.2.2.6. getStringFromUserInput()

```
int getStringFromUserInput (
            char * string,
            int size )
```

La función getStringFromUserInput() recibe una entrada por teclado y la almacena.

Parámetros

string	Se trata de un string (char ∗) en el cuál se almacenará el la cadena de texto recibida.
size	Entero con la tamaño máximo de la cadena que se desea recibir.

Devuelve

La función devuelve un entero indicando la longitud del string recibido. En caso de error devolverá -1.

4.2.2.7. replace()

```
void replace (
            char * string,
            char c1,
            char c2 )
```

La función replace(), reemplaza la el caracter c1 por c2 en toda la string.

Parámetros

string	Se trata de un string que contiene los caracteres que se quieren reemplazar
c1	Se trata de caracter que se quiere reemplazar por otro
c2	Se trata de caracter por el cuál se va a reemplazar

4.2.2.8. split()

```
char** split (
            char * string,
            const char delimiter )
```

La función split() divide el string recibido en substring utilizando como divisor un delimitador. Ejemplo split("Esto es una prueba", ' ') obtendriamos un array con 4 strings "Esto", "es", "una", "prueba".

Parámetros

string	Se trata de un string (char ∗) que contiene la string que se quiere dividir.
delimiter	Se trata de un caracter (char) que contiene el delimitador por el cual se dividirá el string.

Devuelve

La función split() devuelve un array de strings (char ∗∗) en el cual cada posición del array contiene una de las divisiones del string introducido.

Atención

Es necesario liberar la memoria reservada por la funcion split() del parametro devuelto. Para ello liberaremos la memoria reservada para cada string devuelto en el array y luego liberaremos la memoria del array.

4.2.2.9. storeToFile()

```
int storeToFile (
            char * filename,
            char * data )
```

La función storeToFile() almacena en un fichero los datos proporcionados. Esta función creará el fichero si no existe o lo sobreescribirá en caso de que existiera.

Parámetros

filename	Se trata de un string (char $*$) que contiene el nombre del fichero donde se va a guardar los datos
data	Se trata de un string (char $*$) que contiene los datos que se van a almancenar en el fichero

Devuelve

La función storeToFile devuelve un entero con un valor de 0 en caso de éxito y un -1 en caso de error

4.2.2.10. strCelsiusToStrFahrenheit()

```
int strCelsiusToStrFahrenheit (
            char * celsius,
            char ** fahrenheit )
```

La función strCelsiusToStrFahrengeit() recibe grados celsiud en formato string para transofrmalor en grados Fahrenheit también en formato string.

Parámetros

celsius	Se trata de un string (char $*$) que contiene los grados celsiud a convertir
fahrenheit	Se trata de un puntero a string (char $**$) que contendrá el resultado de la conversión de Celsiud a Fahrenheit

Devuelve

La función strCelsiudToStrFahrenheit() devuelve un entero con valor 0 si la operación a resultado con éxito y un -1 si se produce un error.

Atención

Es necesario liberar la memoria reservada por la función strCelsiusToStrFahrenheit(). Para ello liberaremos la mermoria del puntero a string fahrenheit.

4.2.2.11. stringToFloat()

```
int stringToFloat (
            double * value,
            char * data )
```

La función stringToFloat() convierte una cadena de caracteres en float si es posible.

Parámetros

value	Se trata de un double donde se almacenará el número en caso de éxito;
data	Se trata de un string (char ∗) que contiene los datos que se desean convertir

Devuelve

La función stringToFloat devuelve un entero con un 0 en caso de éxito y un -1 si hay algún error.

4.2.2.12. trim()

```
void trim (
            char * string )
```

La función trim() elimina los espacios en blanco ' ' y otros caracters especiales tanto al comienzo como al final del string recibido.

Parámetros

string	Se trata de un string (char ∗) a la que le queremos quitar los espacios en blanco.

4.3. Referencia del Archivo interfaces.c

En este fichero están definidas las funciones declaradas en interfaces.h.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "interfaces.h"
#include "meteo.h"
#include "helpers.h"
```

Funciones

- int **interactiveMenu** (char ∗city, char ∗date, char ∗unit, int ∗days, int ∗printJSON, int ∗saveJSON)
- int showForecast (struct Meteo ∗forecast, int n_data, char unit)

  La función showForecast() imprime por pantalla la información meteorológica proporcionada.

4.3.1. Descripción detallada

En este fichero están definidas las funciones declaradas en interfaces.h.

4.3.2. Documentación de las funciones

4.3.2.1. showForecast()

```
int showForecast (
            struct Meteo * forecast,
            int n_data,
            char unit )
```

La función showForecast() imprime por pantalla la información meteorológica proporcionada.

Parámetros

forecast	Se trata de un array de estructuras de tipo Meteo (struct Meteo ∗) que contiene la información meteorológica.
n_data	Se trata de un entero que contiene la cantidad de entradas meteorológicas que contiene el array forecast.
unit	Se trata de una char que contiene la unidad en la que se representarán las unidades. Estas pueden ser 'C' para Celsius o 'F' para Fahrenheit.

Devuelve

La función showForecast devuelve un entero con valor 0 si se ha ejecutado con éxito, y un -1 si se ha producido un error.

4.4. Referencia del Archivo interfaces.h

En este fichero están declaradas aquellas funciones que tienen como objetivo interactuar con el usuario de la aplicación.

```
#include "meteo.h"
```

Funciones

- int **interactiveMenu** (char ∗city, char ∗date, char ∗unit, int ∗days, int ∗printJSON, int ∗saveJSON)
- int showForecast (struct Meteo ∗forecast, int n_data, char unit)

  La función showForecast() imprime por pantalla la información meteorológica proporcionada.

4.4.1. Descripción detallada

En este fichero están declaradas aquellas funciones que tienen como objetivo interactuar con el usuario de la aplicación.

4.4.2. Documentación de las funciones

4.4.2.1. showForecast()

```
int showForecast (
            struct Meteo * forecast,
            int n_data,
            char unit )
```

La función showForecast() imprime por pantalla la información meteorológica proporcionada.

Parámetros

forecast	Se trata de un array de estructuras de tipo Meteo (struct Meteo ∗) que contiene la información meteorológica.
n_data	Se trata de un entero que contiene la cantidad de entradas meteorológicas que contiene el array forecast.
unit	Se trata de una char que contiene la unidad en la que se representarán las unidades. Estas pueden ser 'C' para Celsius o 'F' para Fahrenheit.

Devuelve

La función showForecast devuelve un entero con valor 0 si se ha ejecutado con éxito, y un -1 si se ha producido un error.

4.5. Referencia del Archivo main.c

En este fichero se inicializa el programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "meteo.h"
#include "interfaces.h"
#include "helpers.h"
```

Funciones

- int **main** ()

4.5.1. Descripción detallada

En este fichero se inicializa el programa.

4.6. Referencia del Archivo meteo.c

En este fichero están definidas las funciones declaradas en meteo.h.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "meteo.h"
#include "helpers.h"
```

defines

- #define **CITIES_BUFFER_INCREMENT** 5
- #define **METEODATA_BUFFER_INCREMENT** 10

Funciones

- void printMeteoEntry (struct Meteo meteoData)

  La funcion printMeteoEntry() muestra por pantalla la información de la entrada meteorológica recibida.
- int loadData (const char ∗filename, struct City ∗∗cities)

  La función loadData() carga datos desde un fichero de texto en formato csv a un array de ciudades de tipo City (struct City ∗). Para cada entrada de este array de ciudades, se encuantrasn los datos meteorológicos correspondientes a cada ciudad. Las ciudades son ordenadas por orden Alfabético para facilitar su búsqueda.
- struct Meteo customMeteoDataParser (char ∗line)

  La función customMeteoDataParser() recibe una string con los datos Meteorológicos en formato csv y los convierte a una estructura de datos de tipo struct Meteo.
- int binarySearchCities (char ∗city, struct City ∗cities, int numCities)

  La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.
- int binarySearchDates (char ∗date, struct Meteo ∗meteoData, int numDataEntries)

  La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.
- int **getForecast** (struct City ∗cities, int numCities, struct Meteo ∗∗forecast, char ∗city, char ∗date, int days)
- int **meteoDataToJSON** (struct Meteo ∗meteoData, int meteoDataLength, char ∗∗jsonData, char unit)

4.6.1. Descripción detallada

En este fichero están definidas las funciones declaradas en meteo.h.

4.6.2. Documentación de las funciones

4.6.2.1. binarySearchCities()

```
int binarySearchCities (
            char * city,
            struct City * cities,
            int numCities )
```

La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.

Parámetros

city	Se trata de un string (char ∗) que contiene el nombre de la ciudad a buscar
cities	Se trata de un array de estructuras de tipo City (struct City ∗) que contiene las cidudades sobre las que se realizará la busqueda
numCities	Se trata de un entero con el número de ciudades almacenadas

Devuelve

La función binarySearchCities devuelve la posición en el array de la ciudad si la ciudad es encontrada. Si no se ha encontrado devolverá -1.

4.6.2.2. binarySearchDates()

```
int binarySearchDates (
            char * date,
            struct Meteo * meteoData,
            int numDataEntries )
```

La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.

Parámetros

city	Se trata de un puntero a una string (char ∗) que contiene el nombre de la ciudad a buscar
cities	Se trata de un array de estructuras de tipo City (struct City ∗) que contiene las cidudades sobre las que se realizará la busqueda
numCities	Se trata de un entero con el número de ciudades almacenadas

Devuelve

La función binarySearchCities devuelve la posición en el array de la ciudad si la ciudad es encontrada. Si no se ha encontrado devolverá -1.

4.6.2.3. customMeteoDataParser()

```
struct Meteo customMeteoDataParser (
            char * line )
```

La función customMeteoDataParser() recibe una string con los datos Meteorológicos en formato csv y los convierte a una estructura de datos de tipo struct Meteo.

Parámetros

line	Se trata de un string (char $*$) que contiene la string con los datos que se quieren cargar.

Devuelve

La función customMeteoDataParser() devuelve una estructura de datos de tipo Meteo (struct Meteo) con los datos cargados del string line

4.6.2.4. loadData()

```
int loadData (
            const char * filename,
            struct City ** cities )
```

La función loadData() carga datos desde un fichero de texto en formato csv a un array de ciudades de tipo City (struct City $*$). Para cada entrada de este array de ciudades, se encuantrasn los datos meteorológicos correspondientes a cada ciudad. Las ciudades son ordenadas por orden Alfabético para facilitar su búsqueda.

Parámetros

filename	Se trata de un string (char $*$) que contiene el nombre del fichero de texto a cargar.
cities	Se trata de un puntero a un array de estructuras de tipo City (struct City $**$) en el que se van a almacenar los datos cargados por ciudades

Devuelve

La función loadData() devuelve el número de elementos cargados si tiene éxito. En caso de error devolverá -1.

Atención

Es necesario liberar la memoria reservada en la función loadData() del parametro cities. Para ello liberaremos la información para cada entrada meteorológica para cada ciudad y posteriormente liberaremos la memoria reservada para cada ciudad.

4.6.2.5. printMeteoEntry()

```
void printMeteoEntry (
            struct Meteo meteoData )
```

La funcion printMeteoEntry() muestra por pantalla la información de la entrada meteorológica recibida.

Parámetros

meteoData	Se trata de una estructura de datos de tipo Meteo la cual contiene la información a mostrar por pantalla

4.7. Referencia del Archivo meteo.h

En este fichero están declaradas aquellas funciones que están relacionadas con el manejo y procesamiento de los datos relacionados con la meteorología.

Clases

- struct City

  Estructura de datos que contiene la informacíon de la ciudad almacenada.
- struct Meteo

  Estructura de datos que continen la información meteorilógica de una ciudad en una fecha determinada.

Funciones

- void printMeteoEntry (struct Meteo meteoData)

  La funcion printMeteoEntry() muestra por pantalla la información de la entrada meteorológica recibida.
- int loadData (const char ∗filename, struct City ∗∗cities)

  La función loadData() carga datos desde un fichero de texto en formato csv a un array de ciudades de tipo City (struct City ∗). Para cada entrada de este array de ciudades, se encuantrasn los datos meteorológicos correspondientes a cada ciudad. Las ciudades son ordenadas por orden Alfabético para facilitar su búsqueda.
- struct Meteo customMeteoDataParser (char ∗line)

  La función customMeteoDataParser() recibe una string con los datos Meteorológicos en formato csv y los convierte a una estructura de datos de tipo struct Meteo.
- int binarySearchCities (char ∗city, struct City ∗cities, int numCities)

  La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.
- int binarySearchDates (char ∗date, struct Meteo ∗meteoData, int numDataEntries)

  La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.
- int **getForecast** (struct City ∗cities, int numCities, struct Meteo ∗∗forecast, char ∗city, char ∗date, int days)
- int **meteoDataToJSON** (struct Meteo ∗meteoData, int meteoDataLength, char ∗∗jsonData, char unit)

4.7.1. Descripción detallada

En este fichero están declaradas aquellas funciones que están relacionadas con el manejo y procesamiento de los datos relacionados con la meteorología.

4.7.2. Documentación de las funciones

4.7.2.1. binarySearchCities()

```
int binarySearchCities (
            char * city,
            struct City * cities,
            int numCities )
```

La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.

Parámetros

city	Se trata de un string (char ∗) que contiene el nombre de la ciudad a buscar
cities	Se trata de un array de estructuras de tipo City (struct City ∗) que contiene las cidudades sobre las que se realizará la busqueda
numCities	Se trata de un entero con el número de ciudades almacenadas

Devuelve

La función binarySearchCities devuelve la posición en el array de la ciudad si la ciudad es encontrada. Si no se ha encontrado devolverá -1.

4.7.2.2. binarySearchDates()

```
int binarySearchDates (
            char * date,
            struct Meteo * meteoData,
            int numDataEntries )
```

La función binarySearchCities() realiza una busqueda binaria en el array de ciudades.

Parámetros

city	Se trata de un puntero a una string (char ∗) que contiene el nombre de la ciudad a buscar
cities	Se trata de un array de estructuras de tipo City (struct City ∗) que contiene las cidudades sobre las que se realizará la busqueda
numCities	Se trata de un entero con el número de ciudades almacenadas

Devuelve

La función binarySearchCities devuelve la posición en el array de la ciudad si la ciudad es encontrada. Si no se ha encontrado devolverá -1.

4.7.2.3. customMeteoDataParser()

```
struct Meteo customMeteoDataParser (
            char * line )
```

La función customMeteoDataParser() recibe una string con los datos Meteorológicos en formato csv y los convierte a una estructura de datos de tipo struct Meteo.

Parámetros

line	Se trata de un string (char ∗) que contiene la string con los datos que se quieren cargar.

Devuelve

La función customMeteoDataParser() devuelve una estructura de datos de tipo Meteo (struct Meteo) con los datos cargados del string line

4.7.2.4. loadData()

```
int loadData (
            const char * filename,
            struct City ** cities )
```

La función loadData() carga datos desde un fichero de texto en formato csv a un array de ciudades de tipo City (struct City $*$). Para cada entrada de este array de ciudades, se encuantrasn los datos meteorológicos correspondientes a cada ciudad. Las ciudades son ordenadas por orden Alfabético para facilitar su búsqueda.

Parámetros

filename	Se trata de un string (char $*$) que contiene el nombre del fichero de texto a cargar.
cities	Se trata de un puntero a un array de estructuras de tipo City (struct City $**$) en el que se van a almacenar los datos cargados por ciudades

Devuelve

La función loadData() devuelve el número de elementos cargados si tiene éxito. En caso de error devolverá -1.

Atención

Es necesario liberar la memoria reservada en la función loadData() del parametro cities. Para ello liberaremos la información para cada entrada meteorológica para cada ciudad y posteriormente liberaremos la memoria reservada para cada ciudad.

4.7.2.5. printMeteoEntry()

```
void printMeteoEntry (
            struct Meteo meteoData )
```

La funcion printMeteoEntry() muestra por pantalla la información de la entrada meteorológica recibida.

Parámetros

meteoData	Se trata de una estructura de datos de tipo Meteo la cual contiene la información a mostrar por pantalla

Índice alfabético