DEVELOPERS DOCUMENTATION OF RUNNING C PROGRAM

The program is a menu driven C program, using a switch structure to navigate through the menu. The whole program is in a .c file, and uses the 'time.h' and 'stdbool.h' libraries in order to manage time as a structure and using booleans.

The run data is stored in a 'Run' structure, which contains:

- Double length; the length in kilometres of the run
- Struct tm start_time; time structure for the beginning of the run
- Struct tm end_time; time structure for the end of the run
- Double duration; derived property obtained from start and end time
- Struct tm pace; derived property obtained from the duration and length

For storing this Run structures Linked Lists are used.

In the **case 1** of the switch structure data is obtained from the user. This is done with scanf function. The parameters obtained are date (start_time date),start and end time, and length. We assume the start_time date and end_time date are the same. Duration parameter is obtained with 'time_difference' function. Pace is obtained by dividing duration/length, and then converting the double value into MM:SS format.

The data inputed by the user is then saved into a text file 'Running.txt' in format

DD/MM/YYYY HH:MM-HH:MM X.X

The derived properties are not stored in the text file since they can be obtained when they are needed.

In **case 2** the program returns a Run of a date inputed by the user. A List is created from the data in the text file 'Running.txt'. This is made with the 'read_file' function.

If the date inputed is the same as the Run's date, that Run information is printed.

Case 3 works in the same way as case 2, but this time two dates are inputed forming a period of time. The program will return all Runs in that period of time. If the first date inputed is before the second chronologically, an error message is displayed.

Case 4 reads the text file and returns the last's Run pace, with the functions 'last_run_pace' and 'last_run'

The default case appears when non of the possible options are selected, works as an error message.

FUNCTIONS DETAILS

INPUT	OUTPUT	DETAILS
`bool equals_dates(struct tm date1,struct tm date2)`		
Two tm structures	If date1 and date2 are the same date ,returns true. Else return is false	
`bool run_between(Run r,struct tm date1,struct tm date2)`		
A Run structure and two tm dates	If Run's date(r.start_time) is between date1 and date2 return true, else return false	Converts the dates into number of days, and the compares them.
`bool first_date_after_second(struct tm date1,struct tm date2)`		
Two tm structures	If date2 is before date1 returns true, else false.	Converts the dates into number of days, and the compares them.
`bool equals_run(Run r1,Run r2)`		
Two run structures	If two runs are the same run returns true, else false	If the Run's date,time and length are equal, we consider it the same run.
`List *insert_run(List *head, Run run)`		
A List and a Run	Returns the List with the new Run element.	
`Run filter_by_date(List * head,struct tm date)`		
A List and a date	Returns the Run from the list with the date given	If no run is found it exits the program
`void filter_by_period(List* head,struct tm date1,struct tm date2)`		
A list and two dates		Prints the Run found between the two dates. If no run is found it prints that no runs were found. Int i is a counter to check that no runs were found.Every time a run is found i increases.
`List *search_run(List *head, Run run)`		
A list and a run	Returns the Run inputed	
`int time_difference(struct tm end,struct tm start)`		
Two tm struct of time	Returns the difference between times in minutes (int)	end_time – start_time
`void run_into_file(int day,int month,int year,int start_hour,int start_min,int end_hour, int end_min,double length)`		