Developer document

Wallet

Homework project, Basics of Programming

Prepared by
Yahia Mahdi, “WF72QQ”

I. Introduction

This document presents an overview about the wallet application for developers who want to participate in its development.

II. Modules and Header file

1. The main.c function contains code for initial menu and the sub menus, while the Header file contains declarations of all used functions, and some of the structures.
2. Display related functions can be found in “displaying_functions.c”
3. The search function can be found in “search_function.c”
4. Category adding, changing and deleting are in “categories.c”
5. Other functionalities are also grouped into separated files making the code cleaner and easier to deal with. These functions are also mentioned in detail in the next section.

III. Used Functions

1. **void append()**

   Reads all the elements of a new entry then adds the entry to the database

2. **char* enter_category()**

   Allows the user to create a category then checks if the category already exists to not enter it twice

3. **void display()**

   Reads all the elements of the database then it prints them. It also calculates the smallest and the greatest value in the database

4. **void delete_category(char* s)**

   Has a string as a parameter (a category name) then deletes it from the categories database and deletes all the elements of the database associated with it

5. **int change_curerncy()**

   Changes the default currency and change all the values in the database from the previous currency to the new one and returns 0 if the conversion happened successfully and 1 if it failed

6. **char* enter_date()**

   A function to enter dates and then returns the entered date as a string

7. **void change(change_rate* p,char *c)**

   Receives the linked list containing the previous currency in the head and the new currency as parameters to find the change rate then updates the database by converting the values of its elements

8. **void cdserach(char *s,char *d)**

   Takes a string containing the date and the category that we are searching as a parameter then prints all the database elements with the same date and category:

9. **int datescmp (int y1,int m1, int d1 ,int y2, int m2 , int d2)**

   Compares two dates entered as integers then return:

   + 1 if the first date is later.

   + 2 if the second date is later.

   + 0 if they are identical.

10. **float balance()**

    Calculates the total money the user has and returns it as a float

11. **char *ReadAndAppendtoCategories(char *c_entry, int add)**

    Adds categories to the categories file and display its elements

12. **change_rate* deletelist(change_rate* head)**

    *A function that will free all the memory allocated to the linked lists after finishing using them

13. **elem* einsert(elem *head ,char* first,char* second ,char* third , char* fourth, char* fifth)**

    Inserts the elements of the database in a linked list

14. **celem* insertcat(celem *head ,char* cat)**

    Inserts the categories of the categories file in a linked list

15. **elem_for_cc* einsert_for_cc(elem_for_cc* head,char* name, char* type,float value,char* category,char* date)**

    Inserts the elements of the database in a linked list after changing the value using change rates.

16. **void list_delete(elem *p)**

    Deletes the linked list storing the elements of the database

17. **void clist_delete(celem *q)**

    Deletes the linked list storing the categories

18. **elem_for_cc* cclist_delete(elem_for_cc* head)**

    Deleting the linked list holding the elements of the database after conversion after writing them into the file

19. **WriteToCategories**

    write the categories stored in the linked list holding the categories after deletion into the categories file

20. **WriteToDataBase**

    write the elements stored in the linked list holding elements of the modified database into the Database file

21. **ccWriteToDataBase**

    write the elements stored in the linked list holding elements of the modified database after currency conversion into the Database file

IV. Structures

1. **A structure that will store the linked list:**

```
typedef struct change_rate{
    char currency[4];
    float rate;
    struct change_rate *nxt;
} change_rate;
```

2. **A structure that has all the elements of an entry:**

```
typedef struct entry{
char name[30];
char type[8];
char category[30];
float value;
char  date[11];
}entry;
```

3. **A structure to create a linked list storing the categories from the file categories after deletion:**

```
typedef struct celem{
char str[30];
struct celem* next;
} celem;
```

4. **A structure to create a linked list storing the data from the database after deletion:**

```
typedef struct elem{
char string1[30];
char string2[8];
char string3[15];
char string4[30];
char string5[11];
struct elem* next;
} elem;
```

5. **A structure to create a linked list storing the data from the Database after the conversion of currency:**

```
typedef struct elem_for_cc{
char string1[30];
char string2[8];
float value;
char string4[30];
char string5[11];
struct elem* next;
} elem_for_cc;
```

V. Used Files

- categories.txt: used to store the categories used in the program or entered by the user.
- Database.txt: used to store all the incomes and expenses entered by the user.
- Currency: used to store the default currency.

VI. Data Structures:

- P1 A linked list to store the change rate from “HUF” to “EUR” and “USD”.
- P2 A linked list to store the change rate from “EUR” to “HUF” and “USD”.
- P3 A linked list to store the change rate from “USD” to ”HUF” and “EUR”.
- A linked list to store the elements from the database after deletion.
- A linked list to store the elements from the file categories after deletion.
- A linked list to store the element from the database after the conversion of currency.