Programming Techniques Project

TO-DO LIST

STUDENT: POP DIANA-ŞTEFANIA

GROUP: C.EN. 1.2A

YEAR: I

Technical report

1 Problem statement

The aim of the assignment is to create a command line application that will have the functionality of a to-do list. The application will prompt the user to enter chores or tasks, and it will store them in a permanent location. It will then allow the user to enter as many tasks as desired but will stop storing tasks by entering a blank task. Tasks are categorized into different categories which can be interacted with (create, delete, modify, etc.), and have different priorities which can be sorted/displayed after. At user's latitude, the tasks can be displayed entirely or, for example, by category. The application also enables the user to remove a task of choice from the list, to signify it's been completed.

2 Pseudocode

Since the information is stored and operated with via simply linked lists the algorithms used to implement the program are fairly simple, such as:

- Adding a node to a list
- Deleting a node from a list
- Displaying the list/the information contained by the list

ADD ($data$, $head$)
1.$iterator \leftarrow head$
2.**while** $next[iterator]$!= NIL **do**
3. $iterator \leftarrow next[iterator]$
4.$last \leftarrow iterator$
5.$next[last] \leftarrow new$
6.$info[new] \leftarrow data$
7.$next[new] \leftarrow NIL$

This algorithm is used in functions such as *add_task*, *initialize_categories* and *create_category*.

DELETE (*data* , *head*)
▷ Removes a node from a list based on its specified field
1.*iterator* ← *head*
2.*element* ← *head*
3.**while** *next*[*next*[*iterator*]] != NIL **do**
4. **if** *info*[*next*[*iterator*]] = *data* **then**
5. *element* ← *next*[*iterator*]
6. *next*[*iterator*] ← *next*[*next*[*iterator*]]
7. free *element*
8. *iterator* ← *next*[*iterator*]
9.**if** *info*[next[iterator]] = data **then**
10. *element* ← *next*[*iterator*]
11. *next*[*iterator*] ← *next*[*next*[*iterator*]]
12. free *element*

This algorithm is used in functions such as *remove_task* and *delete_category*.

DISPLAY (*head*)
1.*iterator* ← *head*
4.**while** *next*[*iterator*] != NIL **do**
5. **write** *info*[*iterator*]
6. *iterator* ← *next*[*iterator*]
7.**write** *info*[*iterator*]

This algorithm is used in functions such as *display_tasks*, *display_by_category*, *display_by_priority*, *update_list*, *update_categories*.

3 Application design

Architectural overview :

The permanent locations to store the information the program will operate with are two files, namely: **tasks.txt** (to store the tasks, along with their category and priority) and **categories.txt** (to store the categories, this file was created due to multiple operations regarding the category field). Firstly, the information from the files will be read into two different simply linked lists : **tasks** and **categories**. After initializing the lists the program will read the commands provided by the user, and the information from the lists will be transferred back into its permanent location.

Principal functions :

- Initialize list
- Initialize categories
- Read commands
- Update tasks
- Update categories

The **read commands** function will treat every command entered by the user as follows:

- Command 't' : It will prompt the user to enter a task with its specified category and priority until entering the blank task which will not be stored. It will verify if the category is valid(already exist in the categories' list) or else it will create it automatically. Also, it will ask for a priority between **1** and **3** (**1** meaning **"low priority"** and **3** meaning **"most important"**).
- Command 'g': It refers to the category field. The user can choose to create a category (command 'c'), delete a category (command 'e') or modify an existing category (command 'm').
- Command 'd': The display command. It can display all tasks (command 'a') or display by category (command 'g') which will ask the user to enter a category to display the tasks after.
- Command 'p': It refers to the priority field. The user can choose to display all task by increasing (command 'i') or decreasing (command 'd') priority.
- Command 'r': The remove task command. It will ask the user to enter the name of the task to remove from the tasks' list.
- Command 'x': It exits the application.

A list of all the other functions used in the program :

- Add task : It adds a task with all its fields (name, category, priority) to the tasks list.
- Valid category : It searches in the categories' list for a specified category and verifies if it exists in the list.
- Create category : It creates a new category in the categories' list.
- Delete category : It deletes the specified category from the categories' list.
- Modify category : Modifies the name of a category into another in the categories' list.

- Modify task category : It modifies a category name from the category field of the tasks' list.
- Display tasks : Displays all tasks with all their fields from the tasks' list.
- Display by category : Displays all tasks (name, priority) with the specified category.
- Display by priority : Displays all tasks (name, category) with the specified priority.
- Remove task : It removes the task with the specified name from the tasks' list.

References

[1] https://codereview.stackexchange.com/questions/29198/random-string-generator-in-c *Random string generator.*

[2] https://en.wikipedia.org/wiki/Linked_list *Linked lists*

[3] LaTeXproject site, http://latex-project.org/