Mode d'emploi Todolist – Thomas Foliard

Informations générales

Ce programme est utilisé en "Command Line Interface". Je n'ai pas inclus de programme de tests (j'ai cru comprendre qu'il fallait faire comme pendant l'amphi de programmation agile, mais je n'ai pas compris comment faire / que tester exactement), mais le repository présente un fichier de sauvegarde type avec des tâches préremplies.

Il sauvegarde les données des tâches dans un fichier JSON à l'aide de la librairie "JSON for modern C++" par nlohmann : https://github.com/nlohmann/json

Elle est intégrée uniquement grâce au header "json.hpp" présent sur ce repository. Elle est écrite en **C++11**, j'ai donc inclus *–std=c++11* pour compiler mon programme.

Celle-ci permet de modifier les contenus d'un fichier json grâce à un objet nlohmann::json, se comportant le plus possible comme un conteneur STL (ici, utilisé comme un dictionnaire, dont les clés sont les IDs des tâches sous forme de *strings*).

Elle utilise entre autres les librairies <iterator>, <vector>, <string> qui sont utilisées dans mon programme. Toutes les opérations (modification, création, suppression de tâches) se traduisent par des modifications de cet objet, donc je n'ai pas eu à définir moi-même de classes, d'où l'absence de header personnel pour ce projet.

Un fichier config.json indique au programme le fichier JSON dans lequel lire et sauvegarder les tâches. Celui-ci est modifiable afin de pouvoir stocker plusieurs todolists à la fois.

Contenu du dossier Github :

- Ce mode d'emploi
- Deux fichiers C++ contenant le code : maintodo.cpp et todolist.cpp
- json.hpp, header de la librairie JSON utilisée
- Un exécutable todo.o

- Un JSON file.json contenant quelques tâches préremplies
- Un JSON “template” ne contenant aucune tâche empty.json
- Un JSON config.json définissant comme fichier de lecture / sauvegarde file.json

Syntaxe des commandes :

./todo.o (--config fichier.json) action (id de tâche) –options

Exemples :

./todo.o --config "file.json" edit 3 -title "Tâche X" -completion 50

./todo.o create -description texte

--config sert à changer le fichier de lecture / écriture, s’il n’est pas spécifié, il reste celui de l’utilisation précédente du programme.

Actions “create” et “edit”:

“edit” nécessite l’identifiant de la tâche à modifier, mais pas “create”, qui crée l’id automatiquement.

On rajoute ensuite une combinaison quelconque des options ci-dessous pour modifier les attributs de la tâche :

Nom *(inutile pour l’utilisateur)*	Type	Valeur par défaut	Option pour le modifier
`"_Title_"`	string	`"Title not defined"`	-title *+ titre*
`"_description_"`	string	`"Description not defined"`	-description *+ desc.*
`"priority"`	string	`"Medium"`	-priority *+ priorité*
`"comments"`	nlohmann::json::array contenant des strings	[ ]	-comment *+ commentaire* (ajoute le commentaire)
`"date_begin"`	string (provenant de la librairie ctime)	Date à la création de la tâche	Aucune, modifié automatiquement
`"date_end"`	string (provenant de la librairie ctime)	`"TBD"`	-close (met la date à la fermeture)
`"status"`	string	`"Open"`	-close (le statut devient “Closed”)
`"completion"`	string	`"0 %"`	-completion *+ entier entre 0 et 100*

`"subtasks"`	nlohmann::json::array contenant des strings	[ ]	-add_subtask *+ id de la sous-tâche*
`"ParentID"`	string	`"0"` (si la tâche n'a pas de parent)	Aucune, modifié automatiquement

Action "delete" :

Nécessite l'id de la tâche à effacer. Efface aussi toutes les sous-tâches qui lui sont affectées, et son id est effacé de l'attribut "subtasks" de sa tâche mère si elle en a une. On ne peut supprimer qu'une tâche à la fois.

Action "display" :

- Si suivi de l'option "**-all**" : affiche toutes les tâches du fichier, distinguant tâches et sous-tâches
- Sinon, affiche toutes les tâches (et leurs sous-tâches) dont l'id est spécifié ensuite. *(**ex:** todo.o display 1 2 6).* NB : appeler l'affichage d'une sous-tâche n'affichera pas la tâche principale avec.

Les sous-tâches sont décalées vers la droite lors de l'affichage pour pouvoir les distinguer, et sont affichées après leur tâche principale

Gestion des inputs incorrects :

Lorsqu'un nom d'action ou d'option entré n'existe pas, le programme ne renvoie pas d'erreur mais n'effectue rien.

Les autres erreurs renvoient un message. De plus, toute action se déroulant correctement entraîne un feedback sur le terminal, de type "Task ID no. 12 created", ou "Title edited", pour être sûr que toutes les commandes ont été rentrées correctement.