Programmation d'interfaces
TP 1 : Introduction à *GTK* et initiation à la *GLib*

Halim Djerroud (hdd@ai.univ-paris8.fr)

Excercice 1 : Échauffement

1. Reprenez les exemples donnés dans le cours (compiler, exécuter et tester).
2. Créer un `makefile` facilement modifiable que vous pouvez réutiliser dans les autres exercices.
3. Écrire une programme en C qui affiche la version de la *GLib*.
4. Écrire un programme qui demande à l'utilisateur de saisir une valeur et l'affiche à l'écran en utilisant exclusivement les fonctions *GLib* et traiter les eventuelles erreurs possibles.
5. Écrire un programme qui demande à l'utilisateur de saisir une valeur entre *10* et *20*, si la valeur n'est pas comprise entre ces deux bornes, alors retournez la valeur la plus proche. Astuce : utilisez une macro *GLib*

Exercice 2 : Les times

1. Écrire un programme qui tire un nombre aléatoire x entre 5 et 10, puis le programme attend x secondes avant de s'arrêter.
2. Écrire un programme qui affiche un compte a rebours de x secondes, x est donné en paramètre du programme.
3. Écrire un programme qui demande à l'utilisateur de saisir un mot, le programme doit afficher le temps de saisie avant de quitter.

Exercice 3 : Les chaînes de caractères (string)

1. Écrire un programme qui demande à l'utilisateur de réorganiser des mots. Chaque mot est rangé dans une case d'un tableau comme suite :
```
[0] -> "Programmation"
[1] -> "génial"
[2] -> "GLib"
[3] -> "La"
[4] -> "avec"
[5] -> "c'est"
```

2. Dans une boucle demandez à l'utilisateur de rentrer les numéro des mots afin de les réorganiser.

3. À chaque fois que l'utilisateur donne un numéro de mot, le concaténer avec le précèdent pour qu'au final, avoir une phrase dans une variable.
4. Affichez la phrase saisie par l'utilisateur.
5. Comparer cette phrase avec `"La programmation avec GLib c'est génial"` si les deux concordent alors affichez `"Bravo !"` sinon afficher `"Dommage, vous avez perdu !"`

Exercice 4 : Les fichiers

1. Écrire un programme qui demande à l'utilisateur de saisir une chaîne de caractère et qui la sauvegarde dans une fichier texte.
2. Écrire un autre programme qui lit ce fichier et affiche à l'écran son contenu.
3. Écrire un autre programme qui affiche les fichiers et répertories de votre espace personnel `" "`

Exercice 5 : Les structures de données

1. Écrire un programme qui demande à l'utilisateur de saisir les 3 étudiants dans une liste chaînées. Les information à enregistrer sont les suivantes :

```
--
Nom : Doe
Prénom : John
Num_etud : 20201234
```

2. Écrire une fonction qui affiche la liste des étudiants.
3. Écrire une fonction qui affiche le nombre d'étudiants dans cette liste.

Exercices à rendre sur moodle

Ces exercices sont à rendre sur moodle dans les 15 jours suivants. A compter de la date de réception de ce TP.

Attention : Utilisez exclusivement les outils fournis par la `GLib`. L'utilisation d'autres librairies ou la réimplémentation des fonctionnalités déjà existantes dans la `GLib` sera considéré hors contexte.

Exercice 1 : Les structures de données

On souhaite écrire une application de gestion de contacts téléphoniques en ligne de commande. Pour chaque personne on sauvegarde son nom et prénom, peut avoir plusieurs numéros de téléphones. Les informations à stocker pour chaque contacte sont les suivantes :

```
--
Nom : Doe
```

```
Prénom : Jane
Num_1 : 01 02 03 04 05
Num_2 : 01 02 03 04 06
Num_n : 01 02 03 04 09
```

Indication

1. Créez un menu pour gérer toutes les fonctionnalités demandées.
2. Utilisez une table de Hachage pour stocker les contactes pour accélérer la recherche par *nom*, indiquer dans le document la fonction de hachage utilisée.
3. À l'intérieur d'un contact utilisez une liste chaînée pour stocker les numéros de téléphones.
4. Utilise une structure une liste chaînée secondaire (liste de pointeurs) pour stocker les contacte dans l'ordre alphabétique.

Gestion des contacts

1. Ajouter simplement un contact avec saisie au clavier, il faut vérifier si le contact existe (même nom et prénom).
2. Afficher la totalité la liste de contacts par ordre alphabétique.
3. Afficher les contacts un par un, par ordre alphabétique.
4. Rechercher un contact avec une partie du nom ou du prénom, si y a plusieurs possibilités, alors afficher l'ensemble des possibilités.
5. Supprimer / Modifier un contact, ces options doivent être intégrées à la recherche.

Exercice 2 : Programme d'entraînement

Nous souhaitons écrire un programme qui aide les utilisateurs a s'entraîner pour saisir au clavier rapidement. Le programme affiche des mots aléatoires puis demande à l'a utilisateur de saisir au clavier le mot affiché. La liste des mots est fournie (dictionnaire), elle se trouve dans le fichier *liste_francais.txt* sur moodle. La liste des mots doit être chargée au démarrage dans une structure de données appropriée, au démarrage du programme. Le programme doit gérer correctement les erreurs par exemple si le fichier n'existe pas.

Le programme doit accepter trois arguments facultatifs, $-m$ pour indiquer au programme le nombre de mots souhaités durant la séance d'entraînement, si l'argument n'est pas fourni la valeur par défaut sera de 10 (utiliser la boucle principale *GLIB* pour implémenter cette fonctionalité). Le second paramètre permet d'indiquer explicitement un autre dictionnaire (un nom de fichier) que celui par défaut. L'option $-h$ affiche un manuel d'utilisation du jeu.

À chaque proposition d'un nouveau mot le programme doit afficher le nombre de mots restants. Une fois que l'utilisateur à saisir le mot proposé le programme doit affiché si le mot est correct ou pas. Le programme doit aussi afficher le temps de saisie de l'utilisateur pour ce mot en millisecondes.

À la fin de la séance d'entraînement le programme affiche un certain nombre de statistiques.

- Le taux d'erreur en pourcentage (nombre de mots corrects / incorrect).
- Le temps total de la séance
- Le temps moyen de saisie (ne prendre en compte que les mots corrects).
- Le mot (correct) qui a demandé le plus de temps de saisie.
- Le temps moyen de saisie par lettre (ne prendre en compte que les mots corrects).

Les options (bonus)

Ces fonctionnalités ne sont pas obligatoires. Un bonus sera attribué pour les étudiant qui aurons a implémenter des fonctionnalités.

- Sauvegarder les statistiques dans un fichier.
- A la fin de chaque séance d'entraînement, comparer les statistiques avec l'historique. Afficher les records.