	Université de Corse - Pasquale PAOLI	
	Diplôme : Licence SPI 3ème année, parcours Informatique	**2023-2024**
	UE : Programmation impérative avancée **Enseignants : Marie-Laure Nivet & Sakhi Shokouh & Diego Grante**	

TP manipulation de structures de données

Pour tous les exercices **il vous est demandé** :

- de respecter la structure de code proposée en cours via le fichier structureProgrammeC.c.
- d'organiser votre code en librairies (Par exemple listes.h listes.c maintest.c).
- d'utiliser dans les codes le type bool via un #include <stdbool.h>
- d'assurer la lisibilité de vos codes et pour cela de les commenter, les indenter, de choisir avec attention les noms de vos variables et de vos fonctions.
- de réaliserez pour chaque programme ou groupes d'exercices un MakeFile associé, permettant de faire le build de l'exécutable ainsi que le clean à posteriori.
- d'être TRES attentif à la gestion de la mémoire.
- de soigner vos tests.
- de poster vos codes sur un dépôt de code dont vous nous donnerez l'adresse.

Par ailleurs vous **DEVEZ RESPECTER VOS MANIFESTES.**

N'oubliez pas que Copilot et ChatGPT ne sont pas inscrits en L3 informatique, en revanche vous, oui ! C'est donc vous qui devez coder !!!!

Objectifs de ce TP

L'objectif principal de ce TP est d'implémenter et d'utiliser des listes chaînées et une pile. Ce TP vous permettra d'aguerrir vos compétences et vos connaissances au niveau de la manipulation et l'utilisation des structures, des pointeurs, des fonctions d'allocation et de libération mémoire en langage C.

Exercice 1 : Utilisation, ré-utilisation, structuration d'un code manipulant/implémentant d'une liste chaînée d'entiers[1]

Récupérez le code listesTPExo1MainCodeDepart.c disponible sur l'ENT. C'est un programme prévu pour permettre les opérations illustrées par le menu console suivant :

```
1 Creer une liste d'entiers compris entre 0 et 20 de taille aleatoire comprise entre 1 et 20
2 Ajouter un élément entier aléatoire (compris entre 0 et 20) en tête de liste
3 Inserer un élément entier aléatoire dans sa place dans la liste suivant l'ordre croissant des premiers éléments
4 Supprimer l'élément de tête
5 Supprimer tous les maillons d'une valeur donnee
6 Detruire liste
7 Sauver la liste courante en binaire dans le fichier "saveliste.bin"
8 Charger une liste depuis le fichier "savelist.bin"
```

Dans ce code il vous est demandé de :

[1] Inspiré et librement adapté d'un code extrait de l'ouvrage *Langage C. Maîtriser la programmation procédurale (avec exercices pratiques) 2e édition - Frédéric Drouillon*. Consulté le: 18 septembre 2022. [En ligne]. Disponible sur : http://www.eni-training.com/portal/client/mediabook/home

- Compléter la fonction main en invoquant les fonctions appropriées. Lorsque plusieurs implémentations sont disponibles, vous utiliserez la version 2 proposée.
- Commenter tout le code à l'image du code inclus dans la fonction de sérialisation `void sauver_liste(maillon_int* prem) ;`
- Organiser et structurer le code en le rendant modulaire. Vous ne laisserez dans le fichier principal – "votrenomexo1.c" que les fonctions main et menu, vous créerez un fichier "votrenomlisteint.h" contenant le type maillon_int, les déclarations des fonctions utilitaires sur les listes chainées d'entiers, dont le code sera placé dans un fichier "votrenomlisteint.c".
- Ecrire un Makefile (cf. Transparents du cours) permettant l'exécution des tâches de compilation de la partie votrenomlisteint.c d'une part, de la partie "votrenomexo1.c" d'autre part, ou bien de l'ensemble, et enfin de nettoyer votre répertoire en effaçant le fichier binaire « savelist.bin », l'exécutable votrenomexo1, ainsi que le fichier objet votrenomlisteint.o.

Exercice 2 : Structure de données Pile [d'entiers].

On rappelle que la structure de donnée pile fonctionne de la manière suivante :
- On part d'une pile vide
- Les éléments sont systématiquement ajoutés en sommet de pile
- Les éléments sont systématiquement retirés du sommet de pile
- On ne peut retirer d'élément d'une pile vide.

Question 1 – Proposez une base pour l'implémentation du type abstrait Pile (tableau statique, tableau dynamique, liste chainée, liste doublement chainée, liste circulaire, etc.) et justifiez votre choix. Attention ici il ne vous est pas demandé d'écrire du code, seulement de justifier votre choix d'implémentation. listes chainer car exo s'intéresse seulement des insertion et suppression des donnés et non pas la recherche ensuite car plus simple et utilise uniquement la mémoire nécessaire pour stocker les éléments réels de la pile de plus Une liste chaînée permet une allocation dynamique de mémoire, ce qui signifie que la taille de la pile n'est pas fixe

Question 2 – En respectant l'implémentation proposée dans la question 1 écrivez les fonctions permettant de :
- Tester si la pile est vide
- Empiler un nouvel élément de type entier
- Récupérer la valeur de l'élément de sommet de pile
- Récupérer la valeur et enlever l'élément de sommet de pile

Question 3 – En utilisant l'implémentation de Pile d'entier que vous avez proposée via les deux questions précédentes écrivez une fonction qui, à partir d'une liste d'entiers, la renverse et affiche la liste renversée en utilisant une pile.

Exercice 3 : Gestion des horaires de trains

On souhaite réaliser un système permettant de gérer une liste d'horaires de trains. Un horaire comprend une ville de départ, une ville d'arrivée, une heure de départ, une heure d'arrivée et la distance entre les deux villes.
Exemple :

Lille	Paris	08h00	08h59	237
Lille	Lyon	07h00	10h00	709
Lille	Calais	15h19	18h34	110
Paris	Marseille	12h00	18h00	900
Lyon	Marseille	10h07	15h01	450
Lyon	Marseille	11h50	17h00	450

Question 1 – Après avoir lu l'exercice en entier, proposez une structure de données permettant de représenter et de manipuler au mieux une liste d'horaires de trains. Vous veillerez à justifier/expliquer votre choix dans des commentaires inclus dans votre code. Dans toute la suite, on supposera que l'on dispose d'une liste d'horaires stockée dans une variable du type indiqué en réponse à cette question.

Question 2 – Écrire une fonction permettant d'ajouter un nouvel horaire de train à une liste existante en précisant toutes les informations nécessaires à la saisie de l'horaire.

Question 3 – Écrire une fonction permettant d'afficher tous les trains au départ d'une ville donnée.

Question 4 – Écrire une fonction qui détermine le trajet sur lequel la vitesse moyenne est la plus élevée.

Question 5 – Écrire une fonction permettant de trier la liste des horaires par ordre croissant de l'heure de départ.

Question 6 – Écrire une fonction permettant de déterminer le moyen le plus rapide pour aller d'une ville à une autre avec au plus une correspondance, étant données une ville de départ (vd) et une ville d'arrivée (va) l'algorithme doit déterminer s'il est possible d'aller de vd à va directement où en empruntant successivement deux trains. Une correspondance est possible uniquement si l'écart entre l'heure d'arrivée du premier train et l'heure de départ du second train est compris entre cinq minutes et deux heures. Par exemple, dans la liste ci-dessus, la seule possibilité pour aller de Lille à Marseille est de prendre Lille-Lyon (7h00 - 10h00) puis Lyon - Marseille (11h50-17h00).