DS N° 2 - INFORMATIQUE II - DOUBLES DIPÔMES

Calculatrice et documents non autorisés

> Une fonction / procédure peut être écrite en plusieures fonction si besoin.

> Si vous ne parvenez pas à répondre à une question, vous pouvez passer à la suivante en supposant que la réponse précédente a été fournie correctement).

> Les fonctions déjà écrites peuvent être ré-utilisées dans les questions suivantes.

> On ne vous demande pas d'écrire la fonction principale du programme.

> Vos programmes doivent être rédigés en C !

> Pretez attention à la ROBUSTESSE de vos programmes !!!

Exercice 1 (*Echantillons d'un tableau*)

1. Écrire une fonction *creerTableau()* qui ne prend pas de paramètre. Cette fonction va demander à l'utilisateur de saisir la taille d'un tableau d'entiers puis va créer et remplir ce tableau avec des entiers aléatoires entre 1 et 1000 (incluses). Le reste du programme doit pouvoir utiliser le tableau ainsi que sa taille.
2. Écrire une fonction/procédure `afficherRec(...)` qui prend en paramètre un tableau d'entiers et sa taille et qui l'affiche. Ce programme doit être **récursif**.
3. Écrire une fonction/procédure *compterTab* qui prend en paramètre **au moins** :
   — Un tableau d'entier
   — Sa taille
   — Une valeur minimale
   — Une valeur maximale

   et qui retourne le nombre de valeurs du tableau comprises entre le minimum et le maximum passés en paramètre. Cette fonction doit être **récursive terminale**.
4. Implémenter une fonction/procédure *remplirTab* qui va prendre en paramètre **au moins** :
   — Un tableau de base
   — Sa taille
   — Un tableau à remplir
   — Sa taille (que l'on supposera correcte)
   — Une valeur minimale
   — Une valeur maximale

   et va remplir le second tableau avec les valeurs du premier tableau qui sont comprises entre le minimum et le maximum. Cette fonction doit être **récursive**.
5. On suppose que l'utilisateur a choisi deux intervalles $[a, b]$ et $[c, d]$ avec pour seule contrainte $a < b$ et $c < d$ (ex : [10, 40] et [25, 90]).
   Implémenter une fonction/procédure qui va prendre en paramètre un tableau d'entiers, sa taille, ainsi que les 4 entiers indiquant les deux segments mentionnés ci-dessus.
   Cette fonction doit créer et remplir deux tableaux, chacun devant contenir les valeurs du tableau de base contenues dans un segment (par exemple si on a [10, 40] et [25, 90], la fonction va créer un tableau avec les valeurs du tableau de base entre 10 et 40 et un autre avec les valeurs entre 25 et 90). Les tableaux créés ainsi que leurs informations utiles doivent pouvoir être utilisés dans le reste du programme.

. Quel est l'ordre de complexité de la fonction précédente (justifier) ?

Exercice 2 (*Gestion des chaussures de sport dans un magasin*)

Un magasin spécialisé dans les articles de sport souhaite informatiser la gestion de ses chaussures de sport. Un modèle de paire de chaussures est caractérisé par les informations suivantes :

- Marque des chaussures
- Prix des chaussures
- Numero de série des chaussures (entier)
- Nombre de paires de chaussures de ce modèle en stock

1. Définissez une structure nommée `Chaussure` pour stocker les informations sur un modèle de chaussures.
2. Écrire une fonction `Chaussure * creerChaussure()` qui permet de créer une nouvelle paire de chaussures en demandant à l'utilisateur de saisir ses champs. Comme indiqué, cette fonction devrait allouer dynamiquement de la mémoire pour stocker les informations de la chaussure et retourner un pointeur vers cette structure.
3. Implementer une fonction/procédure *creerStock()* qui va demander au gestionnaire du magasin de donner le nombre de modèle de son magasin et va creer un tableau contenant tous les modèles de chaussures. Pour des raisons de gestion de la mémoire, le tableau ne contiendra pas directement des *Chaussures* mais des **pointeurs sur Chaussure**. La fonction doit également permettre de récuprerer les informations relatives au différents modèles stockés grace à la question 1. Enfin, elle devra communiquer au reste du programme le tableau créé ainsi que sa taille.
4. Implementer une fonction/procédure *achat()* qui prend en paramètre le tableau de modèles de chaussures créé à la question précédente, sa taille ainsi qu'un numéro de série. Cette fonction permet de simuler l'achat d'une paire de chaussures du modèle en paramètre et doit donc modifier la valeur du nombre de paires en stock du modèle correspondant au numéro de série passé en paramètre.
5. Quel est la complexité de la fonction précédente ? Comment pourrait-on faire mieux ?

Exercice 3 (*Distance de Levenshtein*)

La distance de Levenshtein est un nombre entier donnant une mesure de la différence entre 2 chaines de caractères. On souhaite coder cette fonction en langage C.

La distance de Levenshtein est déterminée par la formule suivante :

$$\text{levenshtein}(\text{strA}, \text{strB}) = \begin{cases} max(|strA|, |strB|) & si \quad min(|strA|, |strB|) = 0 \\ levenshtein(strA-1, strB-1) & si \quad strA[0] = strB[0] \\ 1 + min \begin{cases} levenshtein(strA-1, strB) \\ levenshtein(strA, strB-1) \\ levenshtein(strA-1, strB-1) \end{cases} & sinon \end{cases}$$

avec les définitions suivantes :

- `strA` et `strB` sont les chaines de caractères que l'on souhaite comparer.
- `|strA|` est la longueur utile de la chaine de caractères `strA`
- `|strB|` est la longueur utile de la chaine de caractères `strB`
- `strA[0]` est la première lettre de la chaine de caractères `strA`
- `strB[0]` est la première lettre de la chaine de caractères `strB`
- `strA-1` est la chaine de caractères `strA` sans sa première lettre
- `strB-1` est la chaine de caractères `strB` sans sa première lettre

1. *Ecrire* la fonction `levenshtein(...)` de manière **récursive non-terminale**.

Annexe : quelques fonctions utiles

string.h

strlen

Prototype : `size_t strlen(char *s);`