Travaux Pratiques N° 3

Les Structures

EXERCICE 1 :

Ecrire un programme en C qui permet de déclarer une structure nommée « ensemble » comportant un entier, un réel et un tableau de caractères, ce programme doit permettre de saisir les composantes de cette structure et de les afficher.

EXERCICE 2 :

Écrire les déclarations d'une variable "unEmp" de type "Employe" qui comporte les champs d'informations suivantes :

- nom et prénom
- numéro d'employé
- numéro d'assurance sociale
- âge
- salaire hebdomadaire
- poste de travail (parmi les postes : analyste, programmeur, opérateur, secrétaire)

EXERCICE 3:

1. Définir un type *Rationnel* composé de deux entiers: un numérateur et un dénominateur.
2. Ecrire une fonction *LireRationnel* qui effectue la lecture d'un rationnel valide. Le rationnel mémorisé aura été simplifié.
3. Ecrire une fonction *SommeRationnel* qui retourne la somme des deux rationnels valides passés en argument. Le rationnel retourné aura été simplifié.

EXERCICE 4 :

On souhaite enregistrer les notes de mathématiques et de physique pour une classe de 35 élèves et calculer, pour chacun d'eux, la moyenne de ses notes. Proposer une structure de données pertinentes et écrivez un programme permettant d'effectuer la saisie des notes puis l'affichage des résultats.

EXERCICE 5:

Un grossiste en composants électroniques vend quatre types de produits :

- des cartes mères (code 1)
- des processeurs (code 2)
- des barrettes mémoire (code 3)
- des cartes graphiques (code 4)

Chaque produite possède une référence (qui est un nombre entier), un prix en euros et des quantités disponibles.

Il est demandé de :

1. définir le type pour représenter un produit
2. d'écrire un programme qui permet à un utilisateur de saisir une commande d'un produit.

L'utilisateur saisit les quantités commandées et les données du produit. L'ordinateur affiche toutes les données de la commande, y compris le prix.

EXERCICE 6:

1. Ecrire la déclaration d'un type Fiche permettant de mémoriser les informations sur un étudiant :
   - son nom ;
   - son prénom ;
   - sa date de Naissance, de type Date ;
   - sa formation, représentée par deux lettres ;
   - s'il est redoublant ou non ;
   - son groupe de TD, représenté par un entier ;
   - ses notes, représentées par un tableau note d'au plus MAXNOTES réels;
   - un entier nbnotes indiquant le nombre de notes valides dans le tableau note.

2. Ecrire les fonctions *LireFiche* et *EcrireFiche* de lecture et d'écriture d'une Fiche. Aucune note n'est entrée par la fonction *LireFiche*.

3. Ecrire une fonction *AjouteNote* qui reçoit une Fiche et ajoute une note

4. Ecrire une fonction *Moyenne* qui reçoit une Fiche et renvoie, la moyenne des notes de l'étudiant.

EXERCICE 7 :

Soit la structure ***article*** définie par :

```
struct article
{ int numero;          //un numéro qui identifie l'article
char nom[20];
int qte_stock;         // quantité disponible en stock
float prix;
} ;
```

1. Ecrivez une fonction, nommée ***SaisieArticle***, qui saisit les champs d'une variable article passé en paramètre.
2. Ecrivez une fonction, nommée ***AfficheArticle***, qui affiche le contenu des champs d'une variable article passé en paramètre
3. Ecrivez une fonction nommée ***SaisieTabArticle***, qui remplit un tableau T de n articles. T et n sont des paramètres de la fonction.
4. Ecrivez une fonction ***AfficheTabStock***, qui affiche les articles de T ayant une quantité en stock >= à une valeur q. T, n et q sont des paramètres de la fonction

5. Ecrivez un programme qui fait appel aux fonctions ***SaisieTabArticle*** et ***AfficheTabStock***

EXERCICE 8 :

Un abonné est caractérisé par :

- son numéro
- son nom
- sa ville

1. Ecrire le code qui définit un abonné,
2. Ecrire une fonction qui permet de vérifier s'il existe dans la liste un abonné dont le nom est N et la ville est V.
3. Ecrire une fonction qui permet de calculer le nombre d'abonnés d'une ville donnée.
4. Ecrire une fonction qui permet de retourner un tableau des villes distinctes des abonnés.
5. Ecrire une fonction qui retourne un tableau d'abonné d'une ville donnée

EXERCICE 9 :

On utilise les structures C suivantes :

```
typedef struct {
 char * nom ;
 char * num_tel ;
 char * lieu ;
 char * date ;
} t_rendez_vous;

typedef struct {
 int nb;
 t_rendez_vous * rdvs;    // Tableau de t_rendez_vous
} t_carnet;
```

Les chaînes de caractères et les rendez-vous sont gérés par allocation dynamique.
Il est demandé de mettre en oeuvre les fonctions d'initialisation et de destruction suivantes :

1. ***rendez_vous_creer (t_rendez_vous * self)***
2. ***carnet_creer(t_carnet * self)***
3. ***rendez_vous_detruire(t_rendez_vous * self)***
4. ***carnet_detruire (t_carnet * self)***

Il est ensuite demandé de programmer les fonctions suivantes pour la gestion du carnet :

1. ***rendez_vous_afficher(t_rendez_vous * self)*** pour l'affichage du rendez-vous qui lui est passée en paramètre.
2. ***carnet_afficher(t_carnet * self)*** pour l'affichage de tous les rendez-vous d'un carnet.
3. ***rendez_vous_saisir (t_rendez_vous * self)*** qui permet la saisie au clavier des données d'un rendez-vous;

4. ***int carnet_ajouter_rendez_vous(t_carnet * self, t_rendez_vous * rdv)*** qui ajoute un rendez-vous dans un carnet; retourne 0 si l'ajout c'est bien passé, 1 sinon.

5. ***carnet_rechercher_rendez_vous(t_carnet * self, char * date, t_carnet * resultat)*** qui recherche tous les rendez-vous pour une date donnée; le résultat est inséré dans un carnet vierge passé en argument;

6. ***carnet_retirer_rendez_vous(t_carnet * self, char * nom, char * date)*** qui retire un rendez-vous du tableau

EXERCICE 10 :

Un livre est caractérisé par :

- son numéro (entier)
- son titre (chaîne de caractère)
- son auteur (chaîne de caractère)
- son éditeur (chaîne de caractère)
- son prix (réel)

1. Ecrire le code qui définit un livre
2. Ecrire une fonction qui permet de retourner le nombre d'exemplaires d'un livre donné.
3. Ecrire une fonction qui permet de calculer le nombre de livres d'un éditeur donné.
4. Ecrire une fonction qui permet de retourner un tableau de livre d'un auteur donné.
5. Ecrire une fonction qui retourne le livre le plus cher.
6. Ecrire une fonction qui permet de calculer la valeur de la bibliothèque (somme des prix des livres).
7. Ecrire une fonction qui permet de retourner une partie de la liste de livres à partir d'une position Deb à une position Fin.