Université Mohammed V – Agdal S. ZITI
Faculté des Sciences – Rabat 2012/2013
Département d'Informatique

Travaux dirigés 7
Programmation en Langage C
SMA3

Exercice 1 :

Dans cet exercice, nous allons manipuler une structure représentant un étudiant qui comprendra son nom, son prénom et une note.

1. Pour la première version de cet exercice, nous supposerons que nous manipulerons un nombre fini d'étudiants représenté par la constante NMAX.

La structure utilisée sera la suivante :

```
typedef struct etudiant {
      char * nom;
      char * prenom;
      int note;
} etudiant;
```

Comme on peut le voir sur cette structure, les champs `nom` et `prenom` sont des pointeurs sur des chaînes de caractères non allouées car on souhaite utiliser juste la mémoire nécessaire pour le nom et le prénom. Il sera donc nécessaire de réaliser une allocation dynamique de mémoire pour les champs `nom` et `prenom`.

On travaillera sur un groupe d'étudiants dont la définition est `etudiant groupe[NMAX] ;`

a. écrire la fonction saisir dont l'en-tête est `void saisir(etudiant *e)` permettant la saisie d'un étudiant ;
b. écrire la fonction afficher permettant l'affichage d'un étudiant passé en paramètre ;
c. écrire la fonction moyenne dont l'en-tête est `int moyenne(etudiant e[])` retournant la moyenne de tous les étudiants.

2. Dans la réalité, le nombre d'étudiants fluctue tout au long de l'année. Nous allons adapter la question précédente avec cette nouvelle contrainte en utilisant pour représenter les étudiants une liste chaînée.

La définition de notre structure `etudiant` sera donc :

```
typedef struct etudiant {
      char * nom;
      char * prenom;
      int note;
      struct etudiant *suivant;
} etudiant;
```

1. écrire une fonction `insere_en_tete` dont l'en-tête est `void insere_en_tete(etudiant ** classe)` permettant d'insérer en tête de la liste chaînée un étudiant. Il faudra également réaliser une allocation dynamique pour la saisie de l'étudiant

a. écrire la fonction `void affichage(etudiant * groupe)` affichant tous les étudiants saisis jusqu'à maintenant ;
b. écrire la fonction moyenne dont l'en-tête est `int moyenne(etudiant * groupe)` retournant la moyenne de tous les étudiants.

Exercice 2 :

1. Ecrire une structure décrivant une carte grise avec les éléments suivants : nom et prénom du propriétaire, numéro d'immatriculation, puissance fiscale, date de mise en service.
Afficher à l'écran l'ensemble des données de la carte grise que vous avez saisie au clavier (le champ date pourra être considéré comme un tableau de 3 entiers).
2. Reprendre la question précédente en utilisant maintenant une structure `Date` pour modéliser la date de mise en circulation. On aura donc 3 champs entiers (`int`) dans cette structure afin de définir le jour, le mois et l'année. Faire de même l'affichage de toutes les données de la carte grise.
3. Ecrire une fonction `comparaison()` qui prend en paramètre deux dates, à l'aide de la structure `Date`, et qui retourne la valeur entière 1 si les dates sont identiques et qui retourne 0 dans les autres cas.

Exercice 3 :

Le but de l'exercice est la réalisation d'un ensemble dont les éléments peuvent être ajoutés et retirés en temps constant.
L'ensemble est représenté par une structure incluant un tableau dont les cellules contiennent les adresses des éléments, ainsi qu'un entier ayant pour valeur le nombre d'éléments stockés.
Chaque élément est une structure constituée d'une chaîne de caractères et d'un entier appelé indexe qui contient l'indice auquel l'adresse de l'élément est placée dans le tableau. C'est cet indexe qui permet de retirer n'importe quel élément de l'ensemble en temps constant, indépendamment du nombre d'éléments déjà stockés.
Vous devez réaliser les fonctions suivantes :

1. Définir les deux structures `element` et `ensemble`
2. Ecrire la fonction `element* creeElem(char* s)` qui crée un nouvel élément contenant une *copie* de la chaîne s et retourne l'adresse de l'élément créé. Cet élément doit être stocké dans un bloc mémoire de la taille appropriée réservé dans le tas par la fonction malloc.
3. Ecrire la fonction `void ajouteElem(ensemble* w, element* e)` qui ajoute l'élément pointé par e dans l'ensemble pointé par w. On suppose que l'élément à ajouté a été préalablement créé avec la fonction creeElem.
4. Ecrire la fonction `void retireElem(ensemble* w, element* e)` qui retire de l'ensemble pointé par w l'élément d'adresse e. On suppose que cet élément est initialement présent dans l'ensemble.
5. Réaliser une fonction `main()` qui crée des éléments contenant les chaînes «aaa», «bbb», «ccc», qui les ajoute à un ensemble initialement vide, puis qui retire l'éléments contenant la chaîne «bbb».