ECE 2019-2020

Algorithmique et programmation structurée – INF101

ING 1 - PARTIEL

1h30 - AUCUN DOCUMENT AUTORISE - PAS DE CALCULATRICE D'AUCUNE SORTE

Lire attentivement l'énoncé avant de commencer. Points donnés à titre indicatif.
Commenter les grandes étapes des programmes. Aucune variable globale n'est autorisée.
Jusqu'à 2 points réservés pour les commentaires et l'indentation.

Exercice 1 : tableau de structures (7 points)

Soit une structure définie ainsi :

```
typedef struct voyage{
        char VilleDepart[30] ;
        char VilleArrivee[30] ;
        int HeureDepart ;
        int MinuteDepart ;
        int HeureArrivee ;
        int MinuteArrivee ;
    }t_voyage ;
```

- (2 points) Ecrire un sous-programme qui saisit les champs d'une structure voyage :
  - Prototype du sous-programme : **void remplir (t_voyage * V) ;**
  - Le nom des villes peut contenir des espaces.
  - La saisie des heures et des minutes est à blinder (heure de 0 à 23 et minute de 0 à 59)

    Afin d'éviter trop de copies de code identique, vous pouvez écrire un sous-programme pour le blindage.
- (3 points) Ecrire un sous-programme qui compte le nombre de voyages qui partent après un certain horaire (heure/minute) et qui vont d'une ville donnée à une autre. Les caractéristiques de ces voyages seront toutes saisies : ville de départ, ville d'arrivée, heure de départ et minutes de départ.
  - Le résultat du décompte sera rendu à l'appelant
  - Prototype du sous-programme : **int compter (t_voyage Tab[100]) ;**
- (2 points) Compléter le programme principal suivant en écrivant les instructions décrites en gras dans les commentaires. Ajouter les déclarations des variables nécessaires au traitement. Vous complèterez le programme directement sur le sujet.

```
int main()
{
        //déclaration du tableau de voyages
        t_voyage prevision[100] ;




        //traitement qui remplit tous les éléments du tableau de voyages




        //appel du sous-programme de la question2 : int compter (t_voyage Tab[100]) ;
        // affichage du résultat obtenu




        return 0 ;
}
```

Exercice 2 : structure (2 points)

Définir une structure ***pays***, nommé ***t_pays***, dont les champs sont les suivants :

- Nom du pays : (type chaine de caractères de 30 caractères utiles)
- Continent : (type chaine de caractères de 20 caractères utiles)
- Superficie : (type entier)
- PIB (Produit intérieur Brut) : (type entier)
- Densité (Nombre d'habitants au km^2) : (type réel)
- Tableau de 5 villes : (type chaine de caractères de 15 caractères utiles)

Exercice 3 : chaînes de caractères, fichiers textes et tirage aléatoire (8 points)

- Ecrire un sous-programme qui charge un fichier contenant 100 mots (chaînes de caractères sans espaces) de 25 caractères maximum. Les mots seront stockés dans un tableau (tableau de 100 chaînes de 25 caractères utiles maximum). Le nom du fichier est reçu en paramètre. (3 points)
- Ecrire un sous-programme qui recherche un mot dans un tableau (de 100 chaînes de 25 caractères utiles maximum). Le mot et le tableau sont reçus en paramètres. Le sous-programme retourne 1 si le mot est trouvé, 0 sinon. (2 points)
- Ecrire un programme principal qui (3 points) :
  - Charge un fichier de mots appelé « dico.txt » dans un tableau.
  - Saisit un mot (sans espace). Recherche si par l'utilisateur est présent dans le tableau en appelant le sous-programme de recherche (question précédente). Affiche si le mot est trouvé ou non.
  - Tire aléatoirement un mot dans le tableau et l'affiche.

Exercice 4 : matrice d'entiers et fichiers texte (3 points)

- Ecrire un sous-programme qui reçoit en paramètre une matrice de 10*20 entiers ainsi qu'une valeur entière x. Le sous-programme compte le nombre de valeurs dans la matrice qui sont inférieures ou égales à x et sauvegarde ces valeurs dans un fichier texte appelé «sauv.txt».

Annexes

LISTE DE FONCTIONS DE LA LIBRAIRIE string.h UTILES

char *strcat(char *dest, const char *src);	concatène la chaîne src à la suite de dest
int strcmp(const char *, const char *);	compare deux chaînes lexicalement
char *strcpy(char *toHere, const char *fromHere);	copie une chaîne de caractères d'une zone à une autre
size_t strlen(const char *);	retourne la longueur d'une chaîne caractères