Programmation C

Durée : 2h, aucun document autorisé.

1 Tri fusion

Dans cet examen, nous nous intéressons à la programmation d'un algorithme de tri appelé tri-fusion. Nous utiliserons cet algorithme pour trier par ordre croissant un tableau d'entier.
La procédure implémentée par cette méthode de tri est très simple et se décrit récursivement de la manière suivante :

1. découper le tableau à trier en 2 sous-tableaux (partie gauche et partie droite) de même taille (si le tableau à trier est de taille impaire alors il est coupé en deux sous-tableaux dont les tailles diffèrent de un) ;
2. trier les deux sous-tableaux ;
3. fusionner les deux sous-tableaux triés pour obtenir la version triée du tableau orginal.

La récursivité se trouve au point 2, puisqu'on y fait appel à la fonction de tri. Le tri est particulièrement simple lorsque les listes sont de taille 1, car il il n'y a alors rien à faire (information utile pour la programmation de l'algorithme de tri proprement dit).

Le programme à réaliser s'articulera autour du fichier header fusion.h donné ci-dessous, qui définit la structure de donnée tableau :

```
#ifndef __FUSION__
#define __FUSION__

typedef struct{
        int n;
        int *coeff;
} tableau;

/* cree un pointeur sur un tableau dont la taille */
/* est specifiee en parametre                     */
tableau *cree_tableau(int taille);

/* cree un pointeur sur un tableau dont les elements */
/* sont copies du tableau pointe par t               */
tableau *copie_tableau(tableau *t);

/* cree un pointeur sur un tableau dont les elements          */
/* sont les elements d'indice s a e-1 du tableau pointe par t */
tableau *sous_tableau(tableau *t, int s, int e);

/* libere la memoire utilisee par t */
void     detruit_tableau(tableau *t);

/* fusionne les elements de t1 et t2 (en conservant l'ordre) et */
/* les place dans le tableau pointe par le parametre 'final'.   */
/* renvoie 'final'                                              */
tableau *fusionne_tableau(tableau *final, tableau *t1, tableau *t2);

/* effectue le tri du tableau t, qui est modifie. renvoie t */
tableau *tri_tableau(tableau *t);

#endif
```

Répondre aux questions suivantes :

1. Qu'est-ce qu'un fichier header/en-tête ? A quoi sert ce type de fichier ?
2. A quoi servent les directives **#ifndef**, **#define** et **#endif** ?
3. Quelle est la taille mémoire occupée par un objet de type tableau ? Expliquer.
4. Ecrire la fonction cree_tableau (**int** taille).
5. Ecrire la fonction copie_tableau (tableau *t).
6. Ecrire la fonction sous_tableau (tableau *t, **int** s, **int** e).
7. Ecrire la fonction fusionne_tableau (tableau * final , tableau *t1, tableau *t2). Les tableaux t1 et t2 sont triés, et le tableau résultant de leur fusion, le tableau 'final', est également trié. On suppose que ce tableau fait la taille nécessaire.
8. Ecrire la fonction tri_tableau (tableau *t). Cette fonction implémente simplement la stratégie décrite ci-dessus (découpage du tableau en 2, tri des sous-tableaux et fusion des sous-tableaux triés). Attention aux allocations mémoire intempestives.
9. Créer des fichiers es.h et es.c qui fournissent une fonction **void** affiche_tableau (tableau *t) permettant l'affichage d'un tableau à l'écran.
10. Ecrire un programme principal appelé trifusion .c qui demande à l'utilisateur d'entrer un tableau manuellement (après en avoir spécifié la taille), trie le tableau puis affiche le tableau trié.
11. Ecrire un makefile pour la compilation de votre programme.

2 Un peu de réflexion...

Supposez que l'on vous demande de réaliser un programme permettant de détecter automatiquement dans une photographie (numérisée) si elle contient un lion ou non. Expliquez en une dizaine de lignes, éventuellement agrémentées de schémas, comme vous procéderiez (modèle utilisé, description de son fonctionnement, représentation des photographies sous la forme de vecteurs, etc).