Examen

19 décembre 2019 — Durée 2h

Document autorisé : **Mémento C** vierge de toute annotation

On utilise dans ce sujet un ensemble de paquetages, permettant de lire et écrire des listes de *personnes*. Une *personne* (paquetage `personne`) est représentée par son nom (une chaîne de caractères), et sa date de naissance (type `date`, défini dans le paquetage `dates`).

On a aussi à notre disposition des fonction de gestion de dates, et une fonction permettant de filtrer une liste de personnes à partir d'une date d'anniversaire.

La liste des paquetages ainsi que chaque fichier d'en-tête contenant leur spécification se trouve en annexe (l'implémentation n'est pas nécessaire pour les exercices de ce sujet).

Exercice 1. (1 pt) Écrire la fonction `dates_egales` du paquetage `dates`.

Exercice 2. (3 pt) En utilisant les paquetages fournis en annexe, écrire un programme C (dans un fichier nommé `anniversaire.c`) qui prend en argument de la ligne de commande deux noms de fichiers, et qui :

1. lit une liste de personnes dans le premier fichier ;
2. récupère la liste de ces personnes dont c'est l'anniversaire à la date d'exécution du programme ;
3. écrit dans le deuxième fichier la chaîne «Bon anniversaire !», suivie de la liste de ces personnes.

Exercice 3. (1 pt) Donner un exemple d'exécution du programme `anniversaire` : contenu du ou des fichiers d'entrée, ligne de commande pour l'exécution, contenu du ou des fichiers résultats.

Exercice 4. (1 pt) Dessiner un schéma de dépendances entre le programme de l'exercice 2 et les différents paquetages.

Exercice 5. (2 pt) Écrire un Makefile permettant de compiler le programme de l'exercice 2. L'exécution de la commande `make` sans argument doit générer un exécutable nommé `anniversaire`.

Exercice 6. (2 pt) Décrire le domaine de validité du programme `anniversaire`.

Exercice 7. (2 pt) Donner 3 exemples différents de tests de robustesse pour ce programme.

Exercice 8. (4 pt) Décrire un jeu de tests fonctionnels pour ce programme. Donner pour chaque cas la ou les propriétés du test.

Exercice 9. (4 pt) Écrire un programme de génération aléatoire de listes de personnes. Ce programme prend en paramètre (en argument de la ligne de commande) le nombre de fichiers tests à générer. Ces fichiers produits doivent pouvoir être lus sans erreur par le programme `anniversaire`. Les listes produites doivent avoir un nombre variable de personnes (nombre compris dans l'intervalle $[0, \texttt{NMAX}]$), de dates de naissances variables (le nom peut être constant).

Indiquer quelle(s) propriété(s), parmi celles décrites dans l'exercice 8, sont vérifiées par les fichiers tests produits par votre programme de génération aléatoire.

NB : pour générer plusieurs fichiers numérotés de 1 à N, on peut construire une chaîne de caractère composé du préfixe `"test"` suivi de l'entier `i`. Pour cela on peut déclarer la chaîne de caractères :

```
char chaine[10];
```

Puis utiliser la fonction `sprintf(chaine, motif, ...)`, qui va écrire le motif dans la chaîne. Par exemple :

```
sprintf(chaine, "test%d", i);
```

Annexes

Paquetage dates : contenu du fichier dates.h

```
#ifndef _DATE_H_
#define _DATE_H_

#include <stdbool.h>

/* Type date */
typedef struct {
  // Jour : entier sur l'intervalle [1,31]
  int jour;
  // Mois : entier sur l'intervalle [1,12]
  int mois;
  // Année : entier
  int annee;
} date;

/* Renvoie la date courante */
date date_courante();

/* Paramè tres : d1, d2 de type date
   Renvoie vrai si d1 et d2 sont égaux
 */
bool dates_egales(date d1, date d2);

/* Paramè tre :
   d1, d2 de type date
   renvoie vrai si d2 est une date anniversaire de d1 (même jour, même mois)
 */
bool dates_anniversaire(date d1, date d2);

#endif
```

Paquetage personne : contenu du fichier personne.h

```c
#ifndef _PERSONNE_H_
#define _PERSONNE_H_

#include "dates.h"

/* Type personne */
typedef struct {
  // Nom de la personne
  char nom[20];
  // Date de naissance
  date date_naissance;
} personne;

#endif
```

Paquetage personne_es : contenu du fichier personne_es.h

```c
#ifndef _PERSONNE_ES_H_
#define _PERSONNE_ES_H_

#include <stdio.h>
#include "personne.h"

/* Lit une personne dans le fichier f
   précondition : f doit être ouvert en lecture
   Dans le fichier f, la personne doit être au format
   "<Nom> <date>", sur une même ligne.
   <Nom> est une chaîne de caractè res
   <date> est une date au format JJ/MM/AAAA
   La personne lue est écrite dans p (paramè tre résultat)
*/
void lire_personne(FILE * f, personne * p);

/* Écrit la personne p dans le fichier f
   précondition : f doit être ouvert */
void ecrire_personne(FILE * f, personne p);

#endif
```

Paquetage `liste_personnes` : contenu du fichier `liste_personnes.h`

```c
#ifndef _LISTE_PERSONNES_H_
#define _LISTE_PERSONNES_H_

#include "personne.h"
#include "dates.h"

/* Structure cellule */
typedef struct cellule {
  personne pers;
  struct cellule * suiv;
} cellule;

/* Type liste_personnes : liste de personnes, pointeur vers une
 * structure cellule */
typedef cellule * liste_personnes;

/* Renvoie une liste vide */
liste_personnes liste_vide();

/* Filtre : renvoie la liste des personnes de la liste de départ
 * l_depart pour lesquels la date d est un anniversaire */
liste_personnes filtre_date(liste_personnes l_depart, date d);

#endif
```

Paquetage `liste_personnes_es` : contenu du fichier `liste_personnes_es.h`

```c
#ifndef _LISTE_PERSONNES_ES_H_
#define _LISTE_PERSONNES_ES_H_

#include <stdio.h>
#include "liste_personnes.h"

/* Lit une liste de personnes dans le fichier f
   Précondition : f doit être ouvert en lecture
   Le fichier f contient :
   - un entier N, le nombre de personnes de la liste
   - N lignes au format "<Nom> <date>", ou <Nom> est une chaîne
     de caractè res, et <date> est au format JJ/MM/AAAA
 */
liste_personnes lire_liste_personnes(FILE * f);

/* Écrit la liste de personnes l dans le fichier f
   Précondition : f doit être ouvert en écriture
*/
void ecrire_liste_personnes(FILE * f, liste_personnes l);

#endif
```