INF203 - Travaux pratiques, séance 8
Des joueurs de billes

On reprend nos joueurs de billes. Cette fois, au lieu de déclarer un nombre fixe de joueurs sous forme de variables, on veut pouvoir ajouter autant de joueurs que l'on veut.

1 Gestion de l'ensemble des joueurs

Nous souhaitons écrire un certain nombre de fonctions qui nous permettront de gérer un ensemble J de joueurs, chacun possédant un nom (une chaîne de caractères) et un nombre de billes (un entier). Il s'agit d'un ensemble : deux joueurs distincts ne pourront avoir le même nom.

Pour stocker un tel ensemble, on utilise un tableau de joueurs de taille suffisante (supérieure à $|J|$) et on associe chaque joueur à un entier unique dans $\{0, \ldots, |J| - 1\}$ (son indice dans le tableau). On mémorise également à tout moment le nombre de joueurs connus $|J|$. Lisez le fichier `joueurs.h`.

[a] Quel est le nom du type permettant de représenter un ensemble de joueurs ?
De quelle manière sont représentés les joueurs dans un tel ensemble ?
Quel est le cardinal maximal d'un tel ensemble ?
Y a-t-il une limite à la taille du nom d'un joueur ? ■

Notez bien que le fichier `joueurs.h` contient deux déclarations de types, un certain nombre de prototypes, et une description des fonctions associées à ces prototypes. La définition de ces fonctions se trouve dans `joueurs.c` mais pour l'instant, le corps de ces fonctions est vide et elles ne font donc rien.

Lisez le contenu du fichier `billes.c`. Le comportement attendu de ce programme est le suivant : le programme crée un ensemble de joueurs dont les noms sont donnés en arguments sur la ligne de commande, et pour lesquels les nombres de billes sont tirés au hasard. Il attend alors (en boucle) qu'on tape le nom d'un joueur au clavier et nous donne son nombre de billes ; on tape `q` pour quitter. Voici un exemple de trace d'exécution attendue :

```
./billes Atchoum Dormeur Prof Atchoum
Copie successive des arguments dans l'ensemble : Atchoum - Dormeur -
Prof - Atchoum -
Ensemble de joueurs dans lequel la recherche est faite :
{ Atchoum 42 Dormeur 203 Prof 1 }
Qui voulez-vous ? Prof
1
Qui voulez-vous ? Simplet
-- Joueur absent de l'ensemble
Qui voulez-vous ? Atchoum
42
Qui voulez-vous ? q
```

[b] Compilez le programme contenu dans `billes.c`, sans oublier de mentionner sur la ligne de compilation tous les fichiers `.c` utiles à la compilation (et seulement eux !).

Vérifiez que la compilation s'effectue sans erreur, mais que l'exécution du programme ne fournit pas le résultat attendu. ■

Pour que le programme contenu dans `billes.c` fonctionne, il va falloir compléter la définition des fonctions de manipulation d'un ensemble de joueurs. Complétez le fichier `joueurs.c`. Il est recommandé de compléter les fonctions selon l'ordre indiqué ci-dessous, et de vérifier au fur et à mesure en exécutant le programme `billes` que votre implémentation fonctionne :

- écrivez d'abord `init_joueurs` qui donne au cardinal de l'ensemble la valeur 0 ;
- écrivez ensuite une version de `ajouter_joueur` qui ajoute le joueur à l'ensemble **sans vérifier** qu'il n'est pas déjà présent ;
- puis, écrivez `nombre_joueurs`, `nom_joueur`, `billes_joueur` et `trouver_joueur` ;
- enfin modifiez `ajouter_joueur` pour tester si le joueur est présent ou non dans l'ensemble (par exemple en utilisant `trouver_joueur`).

[c] Joignez le texte de `joueurs.c` à votre compte rendu.
Indiquez tous les jeux de test effectués pour s'assurer que le programme est correct. ■

2 Lecture et écriture d'un ensemble dans un fichier

Le fichier `joueurs_out.h` contient la déclaration d'une fonction qui permettra d'enregistrer nos ensembles de joueurs dans un fichier. En vous basant sur cette déclaration, écrivez le fichier `joueurs_out.c` correspondant.
[d] Joignez le listing de `joueurs_out.c` à votre compte-rendu. ■

Utilisez ces fonctions dans `billes.c` pour enregistrer l'ensemble dans un fichier `Fichier_Joueurs` au moment de quitter le programme.
[e] Quelle ligne de commande tapez-vous désormais pour compiler `billes.c` ?
Quelle autre modification faut-il apporter pour que la compilation se passe correctement ? ■

Sur le même principe, écrivez des fichiers `joueurs_in.c` et `joueurs_in.h` pour lire le contenu d'un ensemble dans un fichier à l'aide d'une fonction `lire_les_joueurs()`. Modifiez `billes.c` pour que, dans le cas où il ne reçoit aucun argument en ligne de commande, il lise l'ensemble dans `Fichier_Joueurs`. N'oubliez pas que le fichier des joueurs respecte un format bien précis, et servez-vous-en pour le lire correctement.

À partir de maintenant, nous allons automatiser la compilation des programmes. Au lieu d'écrire vous-mêmes la ligne de compilation, vous pouvez utiliser la commande `make` qui fera le travail pour vous. Cette commande utilise les informations du fichier `Makefile` pour savoir quoi faire ; ce fichier vous est fourni, il ne vous est pas demandé de savoir l'écrire vous-mêmes.
[f] Lorsqu'on lit l'ensemble des joueurs dans un fichier, que devient ce fichier en sortie du programme ? Comment pouvez-vous le vérifier ? ■

3 Gestion d'un ensemble

Le fichier `gestion.c` contient un programme qui permet à l'utilisateur de lire un ensemble de joueurs avec leurs billes dans un fichier, de modifier les nombres de billes de chaque joueur à l'aide de trois opérations, qui sont définies dans le fichier `operations.c`, puis de sauvegarder l'état des joueurs dans un (nouveau) fichier.
[g] Écrivez la fonction `modifie_billes` du fichier `joueurs.c`, sur laquelle reposent les fonctions de `operations.c`.
Déterminez la ligne de commande à taper pour compiler `gestion.c`. Remarquez que les mêmes fichiers `.c` peuvent participer à la création de deux exécutables différents ! ■

Compilez et exécutez le programme `gestion` en donnant comme fichier de joueurs d'entrée `Fichier_Joueurs`, et comme fichier de sortie un nom (différent) de votre choix, et en n'utilisant que les requêtes G(ain) et P(erte), puis Q(uitter). En vous aidant du texte des fonctions `gain` et `perte`, complétez le fichier `operations.c` en écrivant la fonction `transfert` qui permet de transférer des billes d'un joueur à un autre. Testez cette fonction.

4 Représentation d'un sous-ensemble

Exercice complémentaire :

Le programme `sous_ensembles.c` tel qu'il vous est fourni construit un sous-ensemble de l'ensemble des joueurs : les joueurs ayant moins de 10 billes. Ce sous-ensemble est représenté par le tableau de "booléens" `sous_ens_joueurs`.

Pour mémoriser cet ensemble dans un fichier, on a prévu la fonction `ecrire_les_elements`, à compléter dans `joueurs_out.c`. Le format du fichier à écrire sera le même que celui des fichiers de joueurs déjà lus ou écrits au cours de ce TP : sur une première ligne, le nombre de joueurs, puis, chacun sur une ligne, le nom d'un joueur et le nombre de billes en sa possession. Écrivez cette fonction `ecrire_les_elements` dans le fichier `joueurs_out.c`.

[h] Comment faites-vous pour compter le nombre d'éléments de l'ensemble à écrire dans le fichier ? ■

Ajoutez les directives `#include` au début de `sous_ensembles.c` permettant de lire les fichiers d'entête dont ce programme a besoin.

[i] Donnez la liste des fichiers d'interface que vous avez inclus en tête du programme. ■

Compilez et testez ce nouveau programme avec un fichier d'entrée d'une dizaine de joueurs de votre choix (dont certains avec des comptes inférieurs à 10 billes).

Ajoutez à `sous_ensembles.c`, sur le modèle de `joueurs_faibles`, des fonctions qui calculent le sous-ensemble des joueurs dont le nom commence par `'H'`, le sous-ensemble des joueurs dont le nom fait moins de 5 lettres ... ou tout autre critère de votre choix. Modifiez le programme pour afficher ces sous-ensembles dans des fichiers différents. *Indication : il faudra modifier le nombre d'arguments de la ligne de commande.* Testez !

[j] Joignez votre programme `sous_ensembles.c` complété à votre compte rendu. ■