R2.04 : programmation bas niveau en langage C

TP2 – pointeurs, tableaux, fichiers textes et programmation modulaire

IUTLCO - Département Info – BUT Informatique

Préliminaires :

Récupérer sur Moodle – R2.04 le fichier **Tab foot.zip**.

Décompresser ce fichier afin d'obtenir le dossier **Tab foot** puis ouvrir ce dossier avec VS Code.

But du TP :

Il consiste à manipuler :

- les pointeurs
- les tableaux statiques
- les entrées/sorties de fichiers textes avec *fprintf* et *fscanf*
- les chaînes de caractères
- la programmation modulaire en C

Le programme ne gère pas le fait que le contenu des fichiers textes lus puisse être différent de ce qui est attendu. De même pour les entrées clavier de l'utilisateur.

Ce TP est à déposer sur Moodle à l'issue de la séance.

Explications

Le dossier **Tab foot** contient 6 fichiers :

1. Le programme, composé de 3 fichiers, permet de gérer une équipe de football :
    - Un programme principal définit dans le fichier *main.c*
    - Un module (ou bibliothèque) permettant de gérer une équipe et constitué des fichiers :
        - *equipe.h* définissant la structure de données qui permet de stocker l'équipe et listant les fonctions de manipulation de l'équipe ;
        - *equipe.c* . définissant toutes les fonctions de manipulation d'une équipe.
2. Les fichiers texte *LOSC.txt* et *OL.txt* décrivant les 2 équipes correspondantes
3. Le fichier *Makefile*

Ouvrir le dossier **Tab foot** dans VS Code et regarder chacun de ces fichiers afin d'en comprendre l'organisation et le contenu à l'aide des commentaires. Ne pas hésiter à poser des questions pour être sûr de bien comprendre.

Le fichier *Makefile* sert à simplifier la compilation de programmes constitués de plusieurs modules. En effet, la compilation de ce programme n'est pas possible en un seul appel au compilateur gcc. Il est nécessaire de compiler les modules 1 par 1 puis de les lier en un exécutable. Les commandes à exécuter dans un terminal afin de créer l'exécutable *foot.exe* sont :

```
gcc -c main.c
gcc -c equipe.c
gcc -o foot main.o equipe.o
```

La commande *gcc -c fichier.c* permet de compiler *fichier.c* et de produire le fichier objet compilé *fichier.o*. La commande *gcc -o programme fichier1.o fichier2.o* permet de lier plusieurs fichiers objets en un fichier exécutable appelé *programme*.

Ouvrez un terminal dans VS Code (ou Powershell), positionnez-vous dans le dossier **Tab foot**, puis testez ces 3 commandes et exécuter le programme *./foot*.

Une manière plus simple de procéder est d'utiliser un utilitaire tel que GNU Make qui est déjà installé avec le compilateur gcc. Sous Linux et OSX il est accessible via la commande *make*. Sous Windows suite à l'installation de MinGW il est accessible via la commande *mingw32-make* (si cette commande n'existe pas alors lancer le terminal de MSYS2 qui a servi à installer gcc et exécuter la commande : « `pacman -S mingw-w64-x86_64-make` »).

L'appel à cette commande (*make* ou *mingw32-make*) depuis un terminal positionné dans le dossier **Tab foot** permet de générer le programme en suivant les indications du fichier Makefile qui liste les modules à compiler et la manière de les compiler. Seuls les modules sources ayant été modifiés sont recompilés, ce qui est appréciable lorsqu'un projet est constitué de centaines de modules qui nécessitent plusieurs minutes ou plusieurs heures de compilation.

```
PS D:\Enseignement\2.04 Bas niveau\TP_C\TP2
 mingw32-make
gcc -c main.c
gcc -c equipe.c
gcc -o foot main.o equipe.o
PS D:\Enseignement\2.04 Bas niveau\TP_C\TP2
```

Question 1 – parcours de tableau à l'aide d'un pointeur (pas de [ ])

Pour l'instant l'affichage de l'équipe ne fournit que le nom de cette équipe et le nombre de joueurs.

Compléter le code de la fonction *afficher_joueurs* afin d'afficher les numéros de maillot et les noms et prénoms des joueurs contenus dans le tableau passé en paramètre. Attention, vous devez impérativement utiliser la notation pointeur pour cette question (il ne doit donc pas y avoir de [ ] dans cette fonction). Respecter l'alignement des numéros de maillots comme ci-dessous.

```
 ./foot
LOSC : 24 joueurs
 1 Mickael LANDREAU
16 Barel MOUKO
30 Alexandre OUKIDJA
 2 Mathieu DEBUCHY
 3 Jerry VANDAM
 6 Pape_N'Diaye SOUARE
14 David ROZEHNAL
15 Da_Conceicao EMERSON
18 Franck BERIA
22 Aurelien CHEDJOU
23 Adil RAMI
 4 Florent BALMONT
 5 Idrissa_Gana GUEYE
 7 Yohan CABAYE
10 Ludovic OBRANIAK
21 Arnaud SOUQUET
24 Rio MAVUBA
26 Eden HAZARD
29 Stephane DUMONT
 8 Moussa SOW
 9 Tulio DE_MELO
12 Cedric BASEYA
17 Pierre-Alain FRAU
27 Yao_Kouassi GERVAIS
```

Question 2 – accès à un élément d'un tableau avec un pointeur + passage de pointeur par adresse

Le capitaine de l'équipe va être défini en utilisant un pointeur mais de 4 manières différentes. Le but de cette question est de vous faire manipuler des pointeurs pour l'accès à un tableau et un passage de paramètre un peu plus complexe.

1. Dans la fonction main définir au début les 2 variables :
   ptrjoueur capitaineLOSC1, capitaineLOSC2 ;
   Le type *ptrjoueur* correspond à un pointeur vers un joueur (voir *equipe.h* et l'exemple de *typedef struct* du cours).
   Ajouter ensuite l'instruction permettant de faire pointer *capitaineLOSC1* vers le 17ième joueur du tableau (Rio MAVUBA) sans utiliser de [ ].
   Après cette instruction, vérifier le joueur indiqué par *capitaineLOSC1* en affichant son nom et son prénom.

2. Appeler ensuite la fonction *definir_capitaine2(...)* que vous devrez définir dans les modules *equipe.c* et *equipe.h*, qui prend en paramètre l'équipe ainsi que *capitaineLOSC2* et le modifie afin de le faire pointer également vers le 17ième joueur (Rio MAVUBA). Ce paramètre étant modifié, il doit être passé par adresse.

   De même, l'équipe doit aussi être passée par adresse alors qu'elle n'est pas modifiée, pourquoi ? Parce que si l'équipe est passée par valeur, elle est recopiée dans le paramètre et le capitaine pointera vers la copie qui n'existe plus à la fin de l'appel à la fonction. Après l'appel à votre fonction, dans le main, vérifier le joueur indiqué par capitaineLOSC3.

3. Maintenant essayons de refaire la même chose sans utiliser le type *ptrjoueur*. Dans la fonction *main* définir au début les 2 variables : *joueur *capitaineLOSC3, *capitaineLOSC4 ;*

   Le type *joueur** correspond également à un pointeur vers un joueur tout comme *ptrjoueur*. Comme dans 1, ajouter ensuite l'instruction permettant de faire pointer *capitaineLOSC3* vers le 17ième joueur du tableau (Rio MAVUBA) sans utiliser de [ ]. Rien ne change dans cette instruction par rapport à 1.

   Après cette instruction, vérifier le joueur indiqué par *capitaineLOSC3* en affichant son nom et son prénom.

4. Appeler ensuite la fonction *definir_capitaine4(...)* que vous devrez définir dans les modules *equipe.c* et *equipe.h*, qui prend en paramètre *capitaineLOSC4* et le modifie afin de le faire pointer également vers le 17ième joueur (Rio MAVUBA). Cela ne change rien dans le *main* par rapport à 2 mais par contre dans *equipe.c* et *equipe.h* ça se complique ! Quand un *int* doit être passé par adresse on met *int** dans la définition de la fonction. Ici c'est un joueur* qui doit être passé par adresse donc en paramètre on doit mettre ... Après l'appel à votre fonction, dans le main, vérifier le joueur indiqué par capitaineLOSC3.

Vous avez maintenant compris l'intérêt de déclarer un type pointeur comme *ptrjoueur* qui simplifie l'écriture du code.

```
Capitaine 1 : Rio MAVUBA

Capitaine 2 : Rio MAVUBA

Capitaine 3 : Rio MAVUBA

Capitaine 4 : Rio MAVUBA
```

Question 3 – entrée-sortie clavier et dans un fichier texte

M Capitaine est déçu que le RCL ne soit pas là. Afin de lui permettre de gérer son équipe préférée, il faudrait créer le fichier texte correspondant. La solution la plus simple est d'utiliser un éditeur de texte pour créer le fichier. Mais nous voulons lui laisser l'opportunité de saisir les joueurs lui-même puisqu'il connaît leurs noms par coeur !

En vous inspirant de la fonction *lire_equipe_fichier*, créer la fonction *lire_equipe_clavier* qui interagit avec l'utilisateur afin de lui demander toutes les informations permettant de créer une équipe. Vous aurez besoin de convertir les appels à *fscanf* par des appels à *scanf*. Pour simplifier le code nous supposerons que toutes les chaînes de caractère (nom de l'équipe, nom et prénom de joueur) ne comportent pas d'espace).

Dans le main, créer une variable *nouvEqFoot* et appeler *lire_equipe_clavier* afin de créer une nouvelle équipe qui sera stockée dans cette variable.

Après lecture de la nouvelle équipe, l'afficher pour la vérifier.

```
Lecture d'une nouvelle equipe :
Nom de l'equipe (sans espace) ? TPC
Nombre de joueurs ? 3
Saisir les nom prenom et numero de maillot du joueur 1
? Delignieres Isabelle 10
Saisir les nom prenom et numero de maillot du joueur 2
? Elleuch Wajdi 8
Saisir les nom prenom et numero de maillot du joueur 3
? Rousselle Francois 1
Fin de lecture de l'equipe, merci.

TPC : 3 joueurs
10 Isabelle Delignieres
 8 Wajdi Elleuch
 1 Francois Rousselle
```

Question 4 – entrée-sortie dans un fichier texte

En vous inspirant encore de la fonction *lire_equipe_fichier*, créer cette fois la fonction *ecrire_equipe_fichier* qui va créer le fichier (option « w » pour write dans l'appel à fopen au lieu de « r » pour read), puis remplir le fichier en parcourant l'équipe passée en paramètre. Cette équipe n'étant pas modifiée par cette fonction, il n'est pas utile de la passer par adresse.

Les appels à *fscanf* vont devenir des appels à *fprintf*, et les paramètres écrits ne seront pas passés par adresse mais par valeur car ils ne sont pas modifiés par *fprintf*.

Attention, la présentation du contenu du fichier final doit être la même que celle du fichier LOSC.txt.

Dans le main, suite à la création et l'affichage de la nouvelle équipe, appeler *ecrire_equipe_fichier* en mettant en paramètre de nom de fichier "nouvelleEquipe.txt".

Vérifier la création du fichier et son contenu en l'ouvrant dans VS Code.

Vous pouvez maintenant lire votre équipe au lieu du LOSC en début de main et tester.

Question 5 – chaînes de caractères

La manipulation de chaînes de caractères utilise la bibliothèque *string.h*. Celle-ci définit de nombreuses fonctions de manipulation des chaînes de caractères.

Toutes les fonctions de la glibc (GNU C library) sont décrites dans le manpage 3 sous Unix/Linux. Pour accéder à la page de manuel d'une fonction il suffit d'entrer la commande « man 3 nom_de_la_fonction ».

Si vous n'avez pas accès aux manpages, celles-ci sont disponible sur le web :
https://www.kernel.org/doc/man-pages/ en sélectionnant la section 3.

Ou alors, aller directement voir la documentation de la GNU C Library :
https://www.gnu.org/software/libc/manual/

Nous allons nous intéresser uniquement à 3 fonctions : *strcat*, *strcmp* et *strcpy*.

Dans un premier temps, retrouver la documentation de ces fonctions avec man ou sur le web. Le mot clé *const* signifie que le paramètre de pourra pas être modifié. Le mot-clé *restricted* indique qu'aucun autre pointeur que celui passé en paramètre ne sera utilisé dans la fonction pour accéder à l'objet vers lequel il pointe.

Dans notre programme, suite à la lecture au clavier de la nouvelle équipe, au lieu d'appeler directement *ecrire_equipe_fichier* avec "nouvelleEquipe.txt" en paramètre, nous allons utiliser le nom de l'équipe pour créer un nom de fichier :

1. En début du fichier main.c, inclure la bibliothèque string.h pour avoir accès à ses fonctions. C'est une bibliothèque de la glibc donc à mettre entre < >.
2. Dans la fonction main, créer une variable *nomFicEq* permettant de contenir une chaîne de 20 caractères pour y stocker le nom du fichier.
3. Après la lecture et l'affichage de la nouvelle équipe, utiliser *strcpy* pour copier le nom de l'équipe dans *nomFicEq*.
4. Utiliser strcat pour ajouter à la fin du nom de fichier l'extention ".txt".
5. Modifier l'appel à *ecrire_equipe_fichier* en mettant cette fois *nomFicEq* en paramètre pour le nom de ficher à utiliser.
6. Tester le programme

Question 6 - strcmp

Regarder la documentation de la fonction strcmp.

Ajouter une nouvelle fonction *trouver_maillot_joueur* qui, à l'aide du nom et du prénom d'un joueur, le recherche dans l'équipe à l'aide d'une boucle while et retourne son numéro de maillot ou alors -1 si le joueur n'est pas trouvé.

Dans le main, tester la fonction pour récupérer le numéro de maillot d'un joueur dont vous demanderez le nom et le prénom à l'utilisateur puis afficher le résultat ou un message d'erreur.

```
Recherche d'un numero de maillot :
Nom du joueur ? MOUKO
Prenom du joueur ? Barel
Barel MOUKO porte le numero 16.
```

```
Recherche d'un numero de maillot :
Nom du joueur ? sgsg
Prenom du joueur ? sgsg
le joueur sgsg sgsg n'existe pas dans l'equipe.
```