# Programmation C
# TP5 : Tout depuis le début

Aurélien BOSSARD

Soit la structure suivante :

```
1 typedef struct joueur {
2         char * nom;
3         char * derniers_resultats;
4         int sexe;
5         int date_naissance;
6 }Joueur;
```

Le nom servira à stocker le nom du joueur ou de la joueuse, le champ derniers_resultats à stocker les derniers résultats d'un joueur sous la forme d'une chaîne de caractères composée uniquement de V et de D pour victoire et défaite, les résultats allant, de la gauche vers la droite, du plus ancien au plus récent.

## Exercice 1 : Création d'un nouveau joueur

Écrire une fonction :

```
1 Joueur nouveauJoueur (char * n, int date_naissance, int sexe);
```

qui crée un nouveau joueur dont le nom est `n`, la date de naissance `date_naissance`, et le sexe `sexe` . Attention, il faut copier l'intégralité de `n` dans le nom du nouveau joueur. Pour cela, il faut au préalable préparer la mémoire. Il faut également que la chaîne de caractères qui encode les derniers résultats soit vide (vide signifie tout de même que le caractère de fin de chaîne est présent). Il faut également préparer la mémoire pour cette chaîne.

## Exercice 2 : Liste chaînée de joueurs

Définir une structure liste_joueurs de raccourci Liste_joueurs qui permet d'encoder une liste chaînée de joueurs. Elle ne comprendra que deux champs : l'un nommé joueur et l'autre nommé next. Attention, toute erreur dans le nom des champs ou de la structure elle-même entraînera un 0 à cette question.

## Exercice 3 : Ajout d'une victoire

Écrire une fonction

```
1 int ajout_victoire (Joueur * j);
```

qui ajoute une victoire à la fin de la chaîne de caractères qui représente les derniers résultats du joueur dont l'adresse est passée en paramètre par le pointeur `j`. Si l'ajout d'une victoire à la fin de la chaîne a échoué, la fonction doit renvoyer 0, 1 sinon. N'oubliez pas que pour ajouter une victoire, il faut que la chaîne de caractères soit de taille suffisante pour stocker un caractère de plus.

## Exercice 4 : Ajout d'une défaite

Écrire une fonction

```
1 int ajout_defaite (Joueur * j);
```

qui ajoute une défaite à la fin de la chaîne de caractères qui représente les derniers résultats du joueur dont l'adresse est passée en paramètre par le pointeur `j`. Si l'ajout d'une défaite à la fin de la chaîne a échoué, la fonction doit renvoyer 0, 1 sinon. N'oubliez pas que pour ajouter une défaite, il faut que la chaîne de caractères soit de taille suffisante pour stocker un caractère de plus.

## Exercice 5 : Recherche d'un joueur

Écrire une fonction :

```
1 int recherche_joueur (char * nom, Joueur * tab, int taille_tab);
```

qui renvoie l'index (le numéro de la case) dans le tableau de joueurs `tab` de taille `taille_tab` du joueur dont le nom de est `nom`. Si plusieurs joueurs portent le même nom, la fonction devra renvoyer l'index du premier homonyme. Si aucun joueur n'est trouvé avec ce nom, la fonction devra renvoyer `-1`.

## Exercice 6 : Affichage d'un joueur

Écrire une fonction :

```
1 void affiche_joueur (Joueur j);
```

qui affiche tous les champs du joueur `j`.

## Exercice 8 : Ajout d'un joueur en début de liste

Écrire une fonction :

```
1 Liste_joueurs * ajout_joueur (Joueur j, Liste_joueurs * liste);
```

qui ajoute un élément contenant le Joueur `j` en tête de liste à la liste de joueurs passée en paramètres (cette liste peut être NULL, auquel cas ajouter en tête de liste revient à créer une liste comportant l'unique nouvel élément).

## Exercice 7 : Créer une liste de joueurs sélectionnés

Écrire une fonction :

```
1 Liste_joueurs * creer_liste_jeunes (Joueur * tab, int taille_tab, int date_mir
```

qui crée une liste chaînée de joueurs qui contient tous les joueurs du tableau `tab` de taille `taille_tab` dont la date de naissance est supérieure ou égale à `date_min`.

# Exercice 8 : Supprimer un tableau de joueurs

Écrire une fonction :

```
1 void supprimer_tab_joueurs (Joueur *tab, int taille_tab);
```

qui libère la mémoire du tableau de joueurs `tab` de taille `taille_tab`, qui a été précédemment alloué dynamiquement. Attention à bien libérer tout ce qui a été alloué dynamiquement au sein de ce tableau.

# Exercice 9 : Sauvegarder un tableau de joueurs

Écrire une fonction :

```
1 int ecrit_tab_joueurs (Joueur * tab, int taillle_tab, char * chemin);
```

qui sauvegarde le tableau de joueurs `tab` de taille `taille_tab` dans le fichier dont le chemin est passé en paramètre. Les joueurs sont écrits les uns après les autres. Pour ce faire, chaque champ de chaque joueur doit être écrit l'un après l'autre. D'abord, chaque caractère des deux chaînes de caractères, caractère de fin de chaîne inclus, puis les deux entiers représentant le sexe et la date de naissance. La fonction doit renvoyer 1 si l'écriture a réussi, 0 sinon.