Projet Football

Il s'agit de créer un programme informatique capable de lire des données sur la saison de football en cours et de sortir quelques statistiques. On se focalisera sur la Ligue 1 (1e division professionnelle française) pour laquelle 28 journées ont pu être jouées avant la suspension du championnat due à la crise du coronavirus.

Description du projet

Le programme manipulera les entités suivantes : *Joueur*, *Equipe*, *Match*.
Le synopsis du programme exécutable sera le suivant :

./foot -d <repertoire> [-j buts|buts/match] [-e points|victoires|defaites|nuls|attaque|defense|goalaverage] [-n <nombre>]

-d <repertoire> : les fichiers constituant la base de données se trouvent dans le répertoire *<repertoire>*

-j buts|buts/match : affiche le classement des joueurs par nombre total de buts marqués ou par ratio buts/match

-e points|victoires|defaites|nuls|attaque|defense|goalaverage : afficher le classement des équipes selon le nombre de points, le nombre de victoires, le nombre de défaites, le nombre de matchs nuls, le nombre de buts marqués, le nombre de buts encaissés ou le goalaverage (différence entre buts marqués et buts encaissés). Evidemment pour le nombre de défaites et de buts encaissés, le moins est le mieux.

-n <nombre> : le nombre d'entités à afficher. En cas de plusieurs entités à égalité au sens du critère défini, on affichera toutes les entités à égalité, ce qui pourra amener à dépasser le nombre fixé.

Une victoire vaut 3 points, un nul 1 point et une défaite 0 point.

La base de données doit contenir un fichier par journée de championnat, dont le nom est J<*numéro de la journée*>*.txt*. Chaque fichier contient une ligne par match joué (les matchs reportés ne sont pas répertoriés) avec la structure suivante :

<équipe à domicile (A)>:<équipe à l'extérieur (B)> <nombre de buts équipe A>-<nombre de buts équipe B> <nom buteur A et minute>[/<nom buteur A et minute>]*:<nom buteur B et minute>[/<nom buteur B et minute>]*

Des blancs additionnels, mais pas de tabulation ou de retour à la ligne, peuvent se trouver entre les différents éléments d'une ligne, y compris au début ou à la fin et entre prénom et nom. Le fichier pour la 7e journée vous est fourni, il doit servir de modèle pour tester la robustesse de votre algorithme car il contient toutes les possibilités d'écriture admises.

La convention pour les noms des joueurs consiste à mettre la ou les initiale(s) de leur prénom suivie(s) d'un point puis leur nom. Il peut y avoir des – dans le prénom (exemple : « J-B. ») ou dans le nom (exemple : « Monnet-Paquet ») mais pas de blancs (on écrira Ben Yedder « BenYedder » dans le fichier). Par contre, afin d'harmoniser et de ne pas prendre T. Silva et T.Silva pour 2 joueurs différents, on stockera les noms avec la convention suivante : pas de blanc entre les initiales, un seul blanc après le point et pas de blanc dans le nom. Pour chaque équipe, un joueur particulier aura pour nom *CSC. <equipe>* (*CSC. Nice* pour Nice) et se verra attribuer les buts contre son camp inscrits par les adversaires de l'équipe. Ce joueur devra évidemment être exclu des différents classements.

Constitution de la base de données

N'ayant pu trouver les données nécessaires sous un format facilement exploitable, il va vous revenir de créer la base de données pour ce projet. Vous allez vous repartir la tâche de créer un fichier journée par personne (il n'y en aura pas pour tout le monde vu que seules 28 journées ont pu être jouées). La source d'information est à cette adresse : https://www.lequipe.fr/Football/ligue-1/page-calendrier-resultats/7e-journee. Il faut cliquer sur le score du match pour accéder à son résumé, les buteurs étant quelquefois listés sous le score sur la première image mais ce n'est pas toujours le cas et il vous faudra alors aller chercher l'information dans le résumé.

Critères d'évaluation

L'objectif de ce projet est de montrer que vous avez acquis la maîtrise des concepts de l'orienté objet et du C++ et que vous savez développer un programme de qualité en C++ : bonne exploitation du potentiel du C++, rigueur de l'écriture et robustesse du code, qualité de la présentation en vue de la maintenance et/ou de la réutilisation.
Sera ainsi prise en compte l'utilisation des fonctionnalités du C++ vues en TP, lorsque cela aura sens évidemment.
Le projet ne doit pas donner lieu à production d'une grande quantité de code (en principe moins de 1000 lignes) mais vous êtes attendus sur le côté qualitatif et non quantitatif. A ce titre, les recommandations faites lors du TP 1 prennent tout leur sens.

Tips

Le filtrage des lignes de match peut être simplifié en utilisant la classe *regex* (C++ 2011). Dans ce cas utiliser la méthode *regex_search()* plutôt que *regex_match()* car il vous faudra décomposer l'analyse en 2 parties : d'abord les équipes et le score et ensuite la liste des buteurs lorsque vous connaîtrez le nombre de buts.

Contraintes de travail et éléments d'évaluation

Le projet est à réaliser individuellement.
Un fichier *Makefile* doit être fourni et la production de l'exécutable doit être obtenue avec la commande *make*, sans argument donc, lancée alors qu'on se trouve dans le répertoire contenant le fichier *Makefile*.
Le programme doit compiler avec la commande *g++ -Wall -pedantic-errors* (vous pouvez évidemment ajouter des options) et ne doit pas générer de warnings. Dans la mesure où vous êtes responsable de l'édition du fichier *Makefile*, vous êtes libres d'imposer la norme du C++ qui vous convient.
Dans la mesure où vous disposez de plusieurs jours pour réaliser ce travail, il est attendu que vous preniez le temps de bien tout vérifier avant de le rendre. En particulier :

- une fois le fichier .zip produit, créer un nouveau répertoire, se déplacer dedans, décompresser le fichier, vérifier que le sous répertoire <*Prénom*><*Nom*> a bien été créé, aller dedans, taper *make* et vérifier que ./foot lance bien l'exécutable. Si la procédure ne fonctionne pas sur mon ordinateur, cf. point suivant.
- projet qui ne compile pas => note 0.
- projet qui plante => note <= 5.

Rendu du projet

Vous devrez m'envoyer par email un fichier *.zip* contenant tous vos fichiers source et uniquement vos fichiers source, ainsi que tout fichier indispensable à la compréhension du code et à sa compilation, regroupés dans un même répertoire de nom <*Prémon*><*Nom*>*.zip*. La décompression devra créer le répertoire <*Prénom*><*Nom*> (avec les majuscules) décompresser les fichiers dedans. Il ne devra pas y avoir de sous-répertoires, tout étant « à plat ». Le fichier .zip ne devra pas dépasser 50ko.
La deadline pour rendre le projet est le **dimanche 10 mai à minuit**, date de l'email faisant foi.
Ne pas m'envoyer de lien sur des serveurs distants où je devrais récupérer le code.
Si problème lié au rejet par le gestionnaire de mail de l'UPPA du fichier zip, m'informer afin que l'on recherche une solution alternative (envoyez moi le message de refus du serveur svp).

Extrait du main et sortie possible pour la liste des instances à partir du fichier J07.txt fourni :

```
 unsigned int n=1;
 if (istrm.is_open())
   for (string line ; getline(istrm, line) && line.length()>10 ;) {
     Match m(&L1, &J1);
     m << line;
     char id[9];
     sprintf(id, "07_%02d", n++);
     M1.insert({string(id), m});
   }
 cout << endl << "Liste des equipes :" << endl;
 for (auto it=L1.cbegin() ; it!=L1.cend() ; it++)
   cout << "  " << it->first;
 cout << endl << endl << "Liste des joueurs :" << endl;
 for (auto it=J1.cbegin() ; it!=J1.cend() ; it++)
   cout << "  " << it->first;
 cout << endl << endl << "Liste des matchs :" << endl << endl;
 for (auto it=M1.cbegin() ; it!=M1.cend() ; it++) {
   cout << "Match " << it->first << " : ";
   cout << it->second;
 }
 cout << endl;
```

```
Liste des equipes :
  Amiens  Angers  Bordeaux  Brest  Dijon  Lille  Lyon  Marseille  Metz  Monaco  Montpellier  Nantes  Nice  Nimes
Paris-SG  Reims  Rennes  Saint-Etienne  Strasbourg  Toulouse

Liste des joueurs :
  A. Golovine  A. Souquet  A. Toure  B. Dia  H. Diallo  H. Kamara  L. Remy  M. Cornet  M. Dembele  P. Burner  R.
Alioui  S. Kalu  S. Mendoza  V. Osimhen  W. BenYedder  Y. Adli  Y. Court

Liste des matchs :

Match 07_01 : Dijon 0 - 0 Marseille

Match 07_02 : Monaco 3 - 1 Nice
  Buteur(s) Monaco :  A. Golovine (29)  A. Golovine (74)  W. BenYedder (80)
  Buteur(s) Nice :  P. Burner (54)

Match 07_03 : Toulouse 0 - 2 Angers
  Buteur(s) Angers :  R. Alioui (88)  R. Alioui (95)

Match 07_04 : Nantes 1 - 0 Rennes
  Buteur(s) Nantes :  A. Toure (77)

Match 07_05 : Lille 2 - 0 Strasbourg
  Buteur(s) Lille :  V. Osimhen (43)  L. Remy (64)

Match 07_06 : Amiens 1 - 3 Bordeaux
  Buteur(s) Amiens :  S. Mendoza (3)
  Buteur(s) Bordeaux :  Y. Adli (8)  Y. Adli (46)  S. Kalu (73)

Match 07_07 : Brest 2 - 2 Lyon
  Buteur(s) Brest :  Y. Court (29)  Y. Court (85)
  Buteur(s) Lyon :  M. Dembele (28)  M. Cornet (69)

Match 07_08 : Montpellier 1 - 0 Nimes
  Buteur(s) Montpellier :  A. Souquet (32)

Match 07_09 : Saint-Etienne 0 - 1 Metz
  Buteur(s) Metz :  H. Diallo (18)

Match 07_10 : Paris-SG 0 - 2 Reims
  Buteur(s) Reims :  H. Kamara (29)  B. Dia (94)
```