Mini projet : les échecs

Adrien Piednoël

1 Consignes

Ce projet s'effectue obligatoirement en binôme.

Un rapport de quelques pages vous est demandée. Ce rapport doit comporter les éléments suivants :

- Une introduction qui énonce clairement le sujet, le plan du document, etc ;
- Une partie qui décrit les algorithmes que vous avez utilisés pour faire le mini-projet. Ces algorithmes seront écrits de préférence en langage naturel et/ou via des diagrammes. En particulier, vous pourrez y souligner vos contributions personnelles par rapport au sujet initial;
- Un mode d'emploi du programme qui décrit aussi bien comment le compiler que comment l'utiliser ;
- Une conclusion dans laquelle vous citerez toutes les améliorations que vous pourriez apporter à votre programme.

Les apports personnels contribuent de façon très significatives à une bonne note du projet sachant que le respect du cahier des charges, le code C fonctionnel correspondant et un rapport qui suit les règles énoncées précédemment vous assure au moins la moyenne.

Les règles suivantes sont à respecter lors de l'écriture du programme :

- Mettre des commentaires à bon escient dans le programme (ne pas commenter chaque ligne du programme mais préciser par exemple le rôle des variables) ;
- Choisir des noms de variables explicites ;
- Indenter le programme ;
- Utiliser différentes fonctions pour gérer les différents blocs logiques.

Attention, le programme doit compiler sur les ordinateurs linux de Polytech sans nécessiter l'installation ou le téléchargement de programmes tiers.

Barème :

2pts : Propreté du code

3pts : Description du projet et mode d'emplois

5pts : Fonctions minimales du projet et justifications dans le rapport

10pts : Fonctions avancées de la section « **Pour aller plus loin »**

La note de base est commune pour le binôme. A celle-ci s'ajoute un bonus individuel allant jusqu'à +2pts basée sur les rendus de TPS.

2 Présentation du mini-projet

L'objectif du projet est de développer un jeu d'échecs.

Le programme affiche le plateau de jeu et la couleur du joueur qui doit jouer son coup. Il est demandé au joueur le coup qu'il souhaite jouer. Ce coup est appliqué au plateau et le processus est répété tant que les deux rois sont sur le plateau.

A minima :

- Le plateau doit être correctement affiché dans le terminal. On pourra prendre la convention suivante : roi = K, reine = Q, fou = B, cavalier = N, tour = R, pion = P.
  Pour un joueur les pièces sont écrite en minuscule, pour l'autre en majuscule.
  On obtient la représentation suivante :

```
+-+-+-+-+-+-+-+-+
|r|n|b|k|q|b|n|r|
+-+-+-+-+-+-+-+-+
|p|p|p|p|p|p|p|p|
+-+-+-+-+-+-+-+-+
| | | | | | | | |
+-+-+-+-+-+-+-+-+
| | | | | | | | |
+-+-+-+-+-+-+-+-+
| | | | | | | | |
+-+-+-+-+-+-+-+-+
| | | | | | | | |
+-+-+-+-+-+-+-+-+
|P|P|P|P|P|P|P|P|
+-+-+-+-+-+-+-+-+
|R|N|B|K|Q|B|N|R|
+-+-+-+-+-+-+-+-+
```

- Le programme doit vérifier que à chaque coup, la case de départ jouée contient une pièce du joueur en cours et la case d'arrivée n'est pas une pièce du joueur en cours.
- Le programme s'arrête et affiche le gagnant quand un roi est capturé.

Pour aller plus loin :

- Meilleure vérification de la validité du coup, seul les déplacements légaux du jeu d'échec sont autorisés ; +3pts
- Sauvegarde / chargement d'une partie ; +3pts
- Promotion du pion ; +2pts
- Calcul et affichage de la valeur des pièces ; +1pts
- Détection des situations d'échec ; +3pts
- Gestion du temps, affichage du temps disponible à chaque joueur, défaite en cas de temps dépassé ; +2pts
- IA minimale qui joue des coups légaux contre le joueur +10pts

Les valeurs de points sont données à titre d'ordre de grandeur et peuvent changer en fonction de l'implémentation de l'algorithme et de la justification dans le rapport.

Cette liste n'est pas exhaustive, n'hésitez pas à implémenter de nouvelles fonctionnalités si vous avez des idées.

3 Modélisation du problème

Voici une modélisation proposée du problème, vous êtes libre de représenter les données différemment si vous le souhaitez.

3.1 Structures de données

```
enum piece

{
    VIDE, PION, TOUR, CAVALIER, FOU, REINE, ROI
};
typedef enum piece Piece;

enum couleur
{
    BLANC, NOIR
};
typedef enum couleur Couleur;

struct case
{
      Piece p;
      Couleur c;
} ;
Typedef struct case Case

struct coup
{
      int xFrom;
      int yFrom;
      int xTo;
      int yTo;
} ;
Typedef struct coup Coup

struct partie
{
      Case** plateau;
      Couleur joueur_actif;
};
Typedef struct partie Partie
```

3.2 Fonctions

`Case **créer_plateau()` : renvoie un pointeur vers un plateau initialisé en début de partie
`void affichage(Partie* partie)` : affiche l'état du jeu (plateau, joueur actif)
`Coup proposition_joueur()`: demande au joueur son coup
`bool verifier_coup(Partie* partie, Coup coup)` : vérifie que le coup est jouable
`void appliquer_coup(Partie* partie, Coup coup)` : applique le coup sur le plateau

Attention ! Quelle que soit la fonction que vous créez, celle-ci ne doit jamais spécifier une taille de tableau en argument.
Vous pouvez écrire :

```
Void ma_fonction(tab* monTableau)
```

Vous ne pouvez pas écrire :

```
Void ma_fonction(tab[8] monTableau)
```

Il est conseillé d'utiliser un fichier d'entête (extension *.h) contenant toutes les définitions de vos fonctions.