Projet noté

(3 TP encadrés : semaines 12–14–16)

1 Introduction

On s'intéresse dans ce projet à un jeu de bataille particulier. Il se joue à $n \geq 2$ joueurs et $p \geq 1$ jeux de cartes contenant chacun les 10, valet, dame, roi et as dans les couleurs ♡, ♠, ♢ et ♣. Chaque joueur a un tas de carte en main. Une partie commence en distribuant toutes les cartes (il y en a $20p$) après les avoir mélangées : $n + (20p \bmod n)$ cartes sont mises dans une pioche et les autres sont données en nombre égal à chaque joueur. Une partie est une séquence de coups jusqu'à un état final : soit un seul joueur a encore des cartes en main et c'est le gagnant soit un nombre de coups maximum est atteint.

À chaque coup, les joueurs ayant encore des cartes s'affrontent ; soit J l'ensemble de ces joueurs. Chaque joueur de J prend la carte au sommet de son tas ; soit C l'ensemble de ces cartes. Si un joueur de J a joué une carte plus forte que toutes les autres dans C alors il remporte toutes les cartes de C qui sont remises au fond de son tas. Sinon, il y a une bataille entre les joueurs qui ont les cartes les plus fortes et de même valeur ; soit $B \subseteq J$ l'ensemble de ces joueurs en bataille et soit $C_B \subseteq C$ les cartes que ces joueurs viennent de jouer.

— Les joueurs de J qui ne sont pas dans B ne sont pas en bataille et ont perdu ce coup. Ainsi, les cartes de C qui ne sont pas dans C_B sont mises au sommet de la pioche.

— Chaque joueur de B prend une seconde carte au sommet de son tas ; soit C_2 l'ensemble de ces cartes ; puis une troisième carte au sommet de son tas ; soit C_3 l'ensemble de ces cartes. Si un joueur de B a joué une troisième carte plus forte que toutes les autres dans C_3 alors il remporte toutes les cartes de C_B, C_2 et C_3 qui sont remises au fond de son tas. Sinon les cartes de C_B, C_2 et C_3 sont mises au sommet de la pioche.

— Lors d'une bataille, un joueur peut prendre des cartes au sommet de la pioche s'il n'en a plus dans sa main. Si la pioche ne le permet pas alors ce joueur a perdu.

Le nombre de coups maximum *nmax* est fixé à chaque nouvelle partie. On fixe aussi un nombre m pour renverser la pioche tous les m coups : la carte au sommet se retrouve au fond, *etc.*

2 Types et structures de données

On considère les types suivants dans ce projet.

- Le type énuméré `couleur_carte = {Coeur, Pique, Carreau, Trefle}` représente les couleurs des cartes.
- Le type énuméré `valeur_carte = {Dix, Valet, Dame, Roi, As}` muni de la relation d'ordre `Dix < Valet < Dame < Roi < As` représente les valeurs des cartes.
- Le type `carte` représente une carte possédant une couleur, une valeur et le nom du jeu auquel elle appartient. On a les opérations suivantes :
  — `couleur_carte couleur(car : carte)` : renvoie la couleur d'une carte `car` ;

— `valeur_carte valeur(car : carte)` : renvoie la valeur d'une carte `car` ;

— `chaine nom(car : carte)` : renvoie le nom d'une carte `car` au format `"nom du jeu"."valeur"."couleur"` ;

— construction et destruction d'une carte.

- Le type `jeu_de_cartes` représente un jeu possédant un nom unique et des cartes. On a les opérations suivantes :
  — `bool est_vide(jeu : jeu_de_cartes)` : retourne vrai si le jeu est vide de cartes ; faux sinon ;
  — `carte retirer_carte(jeu : jeu_de_cartes)` : retire une carte du `jeu` puis la renvoie ;
  — `ajouter_carte(jeu : jeu_de_cartes, car : carte)` : ajoute une carte `car` dans le `jeu` ;
  — `chaine nom(jeu : jeu_de_cartes)` : retourne le nom du `jeu` ;
  — `melanger(jeu : jeu_de_cartes)` : mélange aléatoirement les cartes du `jeu` ;
  — `creer(jeu : jeu_de_cartes)` : construction d'un jeu avec une carte de couleur c et de valeur v pour toute couleur c et toute valeur v.
  — destruction d'un jeu.
- Le type `joueur` représente un joueur possédant un nom unique et un tas de cartes dont le comportement est décrit dans l'introduction. On a les opérations suivantes :
  — `entier nb_cartes(jo : joueur)` : retourne le nombre de cartes dans le tas de `jo` ;
  — `chaine nom(jo : joueur)` : retourne le nom de `jo` ;
  — `carte carte_au_sommet(jo : joueur)` : renvoie la carte au sommet du tas de `jo` ;
  — `carte retirer_carte(jo : joueur)` : retire la carte au sommet du tas de `jo` puis la renvoie ;
  — `ajouter_carte(jo : joueur, car : carte)` : ajoute une carte `car` au fond du tas de `jo` ;
  — construction et destruction d'un joueur.
- Le type `pioche` représente une pioche de cartes telle qu'elle est décrite dans l'introduction. Les opérations seront à définir.
- Le type `bataille` représente un jeu de bataille avec des joueurs, des jeux et une pioche. On a au moins les opérations suivantes :
  — `ajouter_jeu(bat : bataille, jeu : jeu_de_cartes)` : ajoute un `jeu` dans `bat` ;
  — `ajouter_joueur(bat : bataille, jo : joueur)` : ajoute un joueur `jo` dans `bat` ;
  — `entier nb_joueurs(bat : bataille)` : retourne le nombre de joueurs dans `bat` ;
  — `distribuer_cartes(bat : bataille)` : distribue toutes les cartes des jeux vers les joueurs et la pioche ;
  — `reconstituer_jeux(bat : bataille)` : déplace toutes les cartes présentes dans les mains des joueurs ou la pioche vers leurs jeux d'origine dans `bat` ;
  — `commencer_une_partie(bat : bataille, nmax : entier, m : entier)` : reconstitue les jeux, distribue les cartes, fixe le nombre de coups maximum `nmax` et fixe le nombre `m` pour renverser la pioche tous les `m` coups ;
  — `jouer_un_coup(bat : bataille)` : joue un coup de la partie en cours dans `bat` comme décrit précédemment dans l'introduction ;

— `entier test_fin_partie(bat : bataille)` : retourne 0 si la partie en cours n'est pas finie (voir introduction) ; un entier positif s'il y a un gagnant ; un entier négatif sinon (il n'y a pas de gagnant et le nombre de coups maximum a été atteint) ;

— `chaine gagnant(bat : bataille)` : retourne le nom du gagnant de la partie s'il en existe un ;

— construction et destruction d'une bataille.

3 Travail

Consignes

— Ce projet doit être réalisé en binômes (1 binôme = 2 personnes).

— Un fichier archive au format `zip` ou `tar.gz` contenant le rapport au format `pdf` et les programmes `C++` sera rendu au plus tard le **27 avril 2018**.

— Ce fichier sera déposé sur `madoc` par **un** (et un seul) membre du binôme dans l'espace devoir correspondant à son groupe de TP.

— Le rapport sera constitué (seulement) des réponses aux questions 1 à 7.

Questions

1. Définir les signatures, les rôles et les préconditions des opérations de la structure de données abstraite représentant les pioches. *(3 points)*

2. Dans quel ordre les cartes sont-elles rangées dans une pioche ? En particulier, une pioche se comporte t-elle comme une pile ou une file ? Argumenter. *(1 point)*

3. Dans quel ordre les cartes sont-elles rangées dans la main d'un joueur ? En particulier, le tas de cartes se comporte t-il comme une pile ou une file ? Argumenter. *(1 point)*

4. Dans quel ordre les cartes sont-elles rangées dans un jeu de cartes ? Est-ce nécessaire d'avoir un ordre particulier ? *(1 point)*

5. Définir la structure de données en mémoire d'un jeu de bataille. *(2 points)*

6. Écrire l'algorithme de l'opération `jouer_un_coup(bat : bataille)` en utilisant les opérations sur les autres types. *(3 points)*

7. Au cours d'une partie, on souhaite savoir où se trouve une carte donnée au moyen d'une table de hachage en retournant le nom d'un joueur si la carte est dans sa main ou le mot `"pioche"` si la carte est dans la pioche. Définir le type des clés C, le type des valeurs V et la fonction $hcode : C \to \mathbb{N}$ calculant le code de hachage de chaque clé. *(1 point)*

8. Programmer les types sous la forme de classes en `C++` en réutilisant les classes standards le plus possible (`list`, `vector`, `stack`, `queue`, *etc.*). La classe `carte` est fournie et ne doit pas être modifiée. *(6 points)*

   Sur un plan opérationnel, les cartes, les jeux, les joueurs et le jeu de bataille doivent être créés dans la mémoire dynamique. Les jeux de cartes et les joueurs sont créés dans la fonction principale puis ajoutés à un jeu de bataille. Les cartes sont créées dans l'appel du constructeur de chaque jeu et détruites dans l'appel du destructeur. Les cartes circulent entre les différentes entités au sein d'un jeu de bataille : les jeux, les joueurs et la pioche.

Elles doivent être remises dans leurs jeux d'origine au début de chaque partie et quand un jeu de bataille est détruit.

Sur un plan organisationnel, les classes `jeu_de_cartes`, `pioche` et `joueur` peuvent êtres développées et testées séparément car chacune d'entre elles dépend seulement de la classe `carte` qui est fournie. La classe `bataille` sera développée ensuite.

Le fichier `prog_projet.tar.gz` sur `madoc` se décompresse dans un répertoire `prog_projet` contenant les fichiers de la classe `carte`, ceux des classes à développer, celui du programme principal `projet.cpp` et un `Makefile`. La commande `make` permet de compiler les fichiers séparément. La commande `./projet` permet d'exécuter le programme. Un exemple de construction d'un jeu de cartes est donné.

9. Une attention particulière sera portée à la qualité des programmes `C++` pour évaluer le travail réalisé en distanciel : organisation des classes, encapsulation, gestion de la mémoire dynamique, mise en œuvre des *headers*, commentaires, *etc. (2 points)*