Introduction à la Programmation

—

Examen partiel sur machine

—

G1: monasse*(at)*imagine.enpc.fr G2: nicolas.audebert*(at)*onera.fr
G3: alexandre.boulch*(at)*onera.fr G4: maxime.ferrera*(at)*onera.fr
G5: laurent.bulteau*(at)*u-pem.fr G6: pierre-alain.langlois*(at)*eleves.enpc.fr

—

10/11/17

—

1 Enoncé

1.1 Jeu de poker

On se propose de coder un jeu de comparaison de mains de poker, qui désigne le vainqueur. Chaque joueur reçoit 5 cartes (une main) parmi le jeu de 52. Chaque carte comprend une valeur (de 2 à 10, 11=valet, 12=dame, 13=roi, 14=as) et une couleur (trèfle, carreau, coeur ou pique). Une combinaison est par exemple deux cartes de même valeur (une paire), ou cinq cartes de même couleur (une couleur). Les combinaisons ont un ordre suivant leur rareté, le joueur ayant la plus forte combinaison remporte la partie. Les parties 1.3 et 1.4 sont largement indépendantes et vous pouvez décider de traiter des questions dans les deux parties.

Il est plus important de livrer un code clair (commenté et indenté) et qui compile sans warning, même s'il ne répond pas à toutes les questions. Pour cela, vérifiez à chaque étape que votre programme compile et se lance correctement.

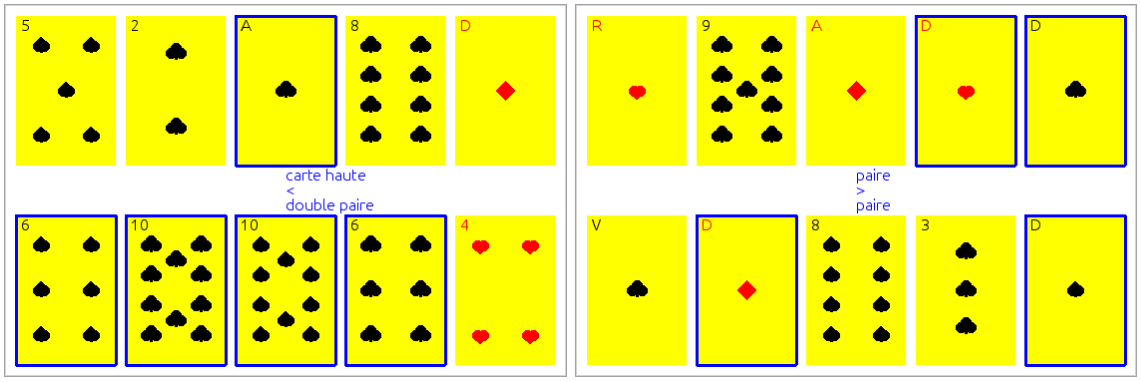

FIGURE 1 – Exemples de deux mains de poker. À gauche, la double paire bat la carte haute, à droite on a égalité de la combinaison (deux dames), la plus haute carte (as) de la main du haut fait la différence.

Créez un projet Imagine++ avec un fichier contenant le `main`. Prenez le `CMakeLists.txt` d'un projet existant et adaptez-le.

1.2 Codage d'une main de poker

1. Dans des fichiers `cartes.h` et `cartes.cpp`, créez une structure `Carte` codant la valeur et la couleur (un nombre entre 0 et 3 pour les 4 couleurs), et faites un operateur == indiquant si ses arguments sont des cartes identiques.

2. Dans les mêmes fichiers, créer une structure `Main` (ne pas oublier la majuscule pour éviter la confusion avec la fonction `main`) qui comprend un tableau de 5 cartes.
3. Écrire une fonction `randMain` qui renvoie une `Main` tirée au hasard. Il s'agit de tirage sans remise, donc une même carte ne peut se retrouver plusieurs fois. La fonction prend donc en argument un tableau des autres mains pour vérifier qu'une carte tirée n'y soit pas déjà présente.
4. Faire afficher à la console (on fera mieux en 1.4 !) le tirage de deux mains avec chaque carte sous une forme abregee, comme `Rcoeur` pour le roi de coeur, `1pique` pour l'as de pique, etc.

1.3 Combinaisons

5. Dans des fichiers séparés, faire une structure `Combi` codant une combinaison obtenue à partir d'une main. Elle comprend une main, les index des cartes constituant la combinaison et leur nombre (2 pour une paire, 3 pour un brelan, 4 pour un carré, 4 pour une double paire, 5 pour un full...), et un score associé à la force de la combinaison (les valeurs de score seront détaillées dans les questions suivantes).
6. Écrire la fonction `haute` qui remplit une `Combi` passée en paramètre avec la carte de plus haute valeur de la main (score 0).
7. Écrire la fonction `paire` qui indique si la main de sa `Combi` passée en paramètre contient une paire de cartes de même valeur et remplit cette `Combi` en conséquence (score 1).
8. Écrire la fonction `suite` qui fait de même si les 5 cartes sont de valeurs consécutives (indépendamment de leur couleur), comme $V, 10, 9, 8, 7$ (score 4).
9. Écrire la fonction `couleur` qui fait de même si les 5 cartes sont de même couleur (ex : tous à carreau), score 5.
10. Écrire la fonction `quinteflush` qui fait de même si on a une suite de 5 cartes de la même couleur (combinaison la plus forte, score 8).
11. Pour coder les autres combinaisons, nous écrivons une fonction `histoVal` qui fait un histogramme des valeurs d'une main. Puisqu'il y a bien moins d'éléments (5) que de valeurs possibles (13), on fait un histogramme compressé composé d'un tableau des valeurs `val` et d'un tableau `histo` du nombre d'occurrences de la valeur (chacun de taille 5). On initialise les tableaux avec des 0, puis pour chaque carte on regarde si sa valeur v a déjà été rencontrée. Si oui, on incrémente le compteur `histo` correspondant, sinon on crée la nouvelle valeur en remplissant une case à 0 de `val` avec la valeur v et mettant son indice `histo` correspondant à 1.
12. Écrire la fonction `carre` qui teste si on a 4 cartes de même valeur (score 7).
13. Écrire la fonction `full` testant la présence de 3 cartes de même valeur et les 2 autres également (score 6).
14. De même pour `doublepaire` composée de deux paires (score 2).
15. De même pour `brelan`, combinaison de 3 cartes de même valeur (score 3).
16. Écrire une fonction `score` qui teste les combinaisons possibles par ordre décroissant de score et stoppe dès qu'une est trouvée. La `Combi` passée en paramètre est alors remplie.

1.4 Affichage

17. Dessiner les couleurs avec des formes élémentaires (trèfle et pique en noir, coeur et carreau en rouge). Ces fonctions prennent des entiers x et y pour indiquer la position de leur origine $(0, 0)$ (voir Figure 2). Pour remplir un polygone, on utilise la fonction `fillPoly` qui prend dans un tableau de $2n$ entiers les coordonnées des sommets, par exemple un triangle :

```
int pt[] = {0,0, 10,0, 0,10};
Imagine::fillPoly(pt, 3, RED);
```

18. Écrire une fonction `dessineCouleur` qui trace une couleur (codée par un entier de 0 à 3) en position (x, y).
19. Écrire une fonction `dessineCarte` qui dessine une carte en (x, y) avec sa valeur et couleur. Une carte est un rectangle jaune de dimension $10 \times$ taille en largeur et $15 \times$ taille en hauteur.

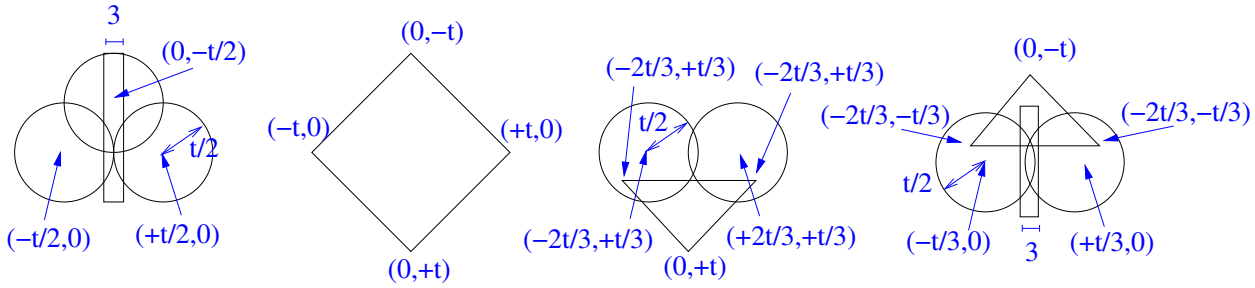

FIGURE 2 – Schéma graphique des couleurs trèfle, carreau, coeur et pique. L'échelle t sera définie par une constante globale `taille` initialisée à 10 ou 20.

20. Écrire une fonction `dessineMain` qui dessine côte à côte toutes les cartes d'une main (avec un écart de 10 pixels) et remplacer l'affichage de la question 4 par cette représentation graphique.
21. Indiquer graphiquement par un encadré bleu les cartes formant la meilleure combinaison de chaque main et afficher le vainqueur (plus haut score).

1.5 Questions bonus

22. Afficher les cartes de façon stylisée, avec autant de fois la couleur représentée que la valeur de la carte si elle est entre 2 et 10 (voir Figure 1 pour la disposition sur la carte).
23. Pour départager deux mains de même score, on compare par ordre de priorité :
    (a) la valeur de la combinaison : valeur de la plus forte paire pour une double paire suivie de la deuxième paire en cas d'égalité, du brelan pour un full, de la plus forte carte pour une suite, couleur ou quinte flush. . .
    (b) En cas d'égalité de la combinaison, on compare les autres cartes par ordre décroissant de valeur. Voir Figure 1.