TP 2 : utilisation de structure simple

L'objectif de ce TP va être d'écrire un programme de gestion du jeu de cartes « *bataille* ». Dans notre version, on jouera avec un jeu de **32 cartes**.

Après mélange et distribution des cartes a deux joueurs, chacun retourne la première carte de son tas. La carte la plus forte gagne la manche et les deux cartes sont mis en dernier dans le tas du gagnant. L'ordre est : As, Roi, Dame, Valet, 10, 9, 8, 7. En cas d'égalité, la couleur la plus forte est gagnante dans l'ordre : Cœur, Pique, Carreau, Trèfle. Le joueur qui ramasse toutes les cartes est déclaré gagnant.

Le programme fait appel à une structure pour représenter une carte dont les membres sont la valeur et la couleur. Les cartes de chaque joueur sont stockées dans un tableau.

Le squelette de programme suivant est fourni :

```
#include /* à compléter */

typedef struct carte {

/* à compléter */

} Carte;


void initjeu(Carte jeu[]) /* remplit le tableau avec les 32 cartes */

{/* à compléter */}

int alea32() /* génère un entier aléatoire compris entre 0 et 31 */

{/* à compléter */}

void permute(Carte *carte1, Carte *carte2) /* permute 2 cartes */

{/* à compléter */}

void melange(Carte jeu[]) /* mélange les cartes */

{/* à compléter */}

void affiche_carte(Carte X) /* affiche une carte (valeur, couleur)*/

{/* à compléter */}

void affiche(Carte jeu[],int N) /* affiche N cartes */

{/* à compléter */}

void distribue(Carte jeu[], Carte jeu1[], Carte jeu2[]) /* distribue les
cartes aux deux joueurs */

{/* à compléter */}

void joue1coup( Carte jeu1[], Carte jeu2[], int* N1, int* N2) /* gère le
résultat d'une bataille (1 coup) */

{/* à compléter */}
```

```
int main() {
	Carte jeu[32];
	Carte jeuJ1[32]; /** cartes du joueur 1 */
	Carte jeuJ2[32];
	int NJ1 = 16; /** Nombre de cartes du joueur 1 */
	int NJ2 = 16;
	srand((unsigned) time(NULL)); /** génération de la graine aléatoire */
	initjeu(jeu);
	melange(jeu);
	distribue(jeu,jeuJ1,jeuJ2);
	printf(" __ Jeu Joueur1 __\n");
	affiche(jeuJ1,16);
	printf(" __ Jeu Joueur2 __\n");
	affiche(jeuJ2,16);
	int NBcoups = 0;
	while ((NJ1 > 0) && (NJ2>0)) {
		joue1coup(jeuJ1, jeuJ2, &NJ1, &NJ2);
		NBcoups++;
		printf("J1 : %d cartes, J2 : %d cartes\n",NJ1,NJ2);
	}
	/* afficher qui gagne et en combien de coups */
	return 0;
}
```

Copier ce code dans votre répertoire **~/TP2** dans le fichier **jeucarte.c**

0. Définition de la structure

Définissez la structure **Carte** permettant de manipuler des cartes de jeu (pensez aux énumérations 😊 !)

1. Initialisation du jeu

Ecrire la fonction **initjeu** qui remplit le tableau jeu avec les 32 cartes ainsi que les fonctions d'affichage d'une carte et du jeu entier.

Vérifier le fonctionnement avant de passer à la suite.

2. Mélange des cartes

Ecrire la fonction **alea32** qui renvoie un entier compris entre 0 et 31 (correspondant à un indice valide du tableau *jeu*).

Ecrire une fonction permettant de permuter deux variables de type C*arte*.

Utiliser ces deux fonctions pour écrire la fonction de mélange des cartes. L'idée est de réaliser un nombre important (50 par exemple) de permutations entre deux cartes tirées au hasard. Vérifier en l'affichant que le jeu est bien mélangé.

3. Distribution des cartes

Ecrire la procédure permettant de distribuer 16 cartes à chaque joueur. On utilisera 1 tableau de 32 Cartes pour chaque joueur. Afficher les 16 cartes de chacun des joueurs et vérifier le bon fonctionnement.

4. Jeu de carte

Voici la partie la plus délicate du programme.

Chaque joueur possède N1 (ou N2) cartes indicées de 0 à N1-1 (ou N2-1). La carte jouée est celle d'indice 0. Une solution consiste à mettre de côté les cartes d'indice 0 de chaque joueur puis à décaler toutes les cartes restantes de chaque joueur vers le bas (cartes 1 à N1-1 (ou N2-1) mises de 0 à N1-2 (ou N2-2)).

Les deux cartes mises de côté sont alors comparées (valeur et éventuellement couleur) et rangées dans le tas du gagnant aux positions N1-1 (pour la carte gagnante) et N1 si le joueur 1 gagne ou N2-1 et N2 si c'est le joueur 2.

Les valeurs de N1 et N2 doivent bien entendu être ajustées en conséquence (+1 pour le gagnant, -1 pour le perdant).

Tester le bon fonctionnement sur quelques coups du jeu en affichant par exemple les cartes restantes des deux joueurs après chaque coup.

5. Jeu final

Compléter le programme principal en écrivant la condition de fin de partie.
Afficher le joueur gagnant et le nombre de coups joués.

On pourra aussi afficher la succession des coups joués.