LIFAP1 – Séquence 1

TP Noté #2 - durée 1h30 mn

Mardi 13 décembre 2022

Sujet A

Consignes

Aucun accès au WEB, aux pages de l'UE, ni à vos anciens TP n'est autorisé.
La note tiendra compte du respect des consignes, de la qualité de la présentation et de la lisibilité du code, des algorithmes, et du bon fonctionnement du programme. **Seules les notions vues en cours devront être utilisées**.
La zone édition contient un début de code, commençant par un commentaire où vous renseignerez vos nom et prénom ainsi que votre numéro d'étudiant.
Une fois le programme terminé et testé (ou à la fin du temps imparti), vous terminerez la session en cliquant sur le bouton EXIT. Vous pourrez sauvegarder votre travail au fur et à mesure en cliquant sur l'icône ✉, indenter votre code en tapant la touche F8, et la touche F9 permettra de compiler le programme, qui s'exécutera automatiquement si la compilation réussit.

Travail à réaliser

Pour réaliser ce TP, vous pourrez utiliser toutes les fonctionnalités de la bibliothèque `string.h`. et vous devrez utiliser les bibliothèques `stdlib.h` et `time.h` pour utiliser la fonction `rand()`.

Nous allons simuler un jeu de cartes basic : la bataille version simplifiée. Dans ce jeu chaque joueur prend une carte dans son jeu et la compare avec celle de son adversaire :

- si elle a une valeur plus grande il conserve sa carte et remporte celle de l'autre joueur,
- si elle a une valeur plus petite il perd sa carte,
- si les deux cartes sont équivalentes chacun conserve sa carte.

Chaque `Carte` sera identifiée par

- sa `valeur` : un entier de 1 à 10 pour les Cartes classiques et de 11 à 13 pour les têtes (valet = 11, dame = 12 et roi = 13),
- et sa "`couleur`" chaine de caractères choisie parmi "`PIQUE`", "`COEUR`", "`CARREAU`", et "`TREFLE`".

Un `Paquet` de `Cartes` sera constitué d'un tableau de `Cartes T` d'au maximum `MAXCARTES Cartes` et du nombre de `Cartes nb` qu'il contient.

1- Définir en langage C/C++ deux constantes nommées `MAXCH` et `MAXCARTES` ayant pour valeurs respectives 10 et 52.
2- Définir les structures `Cartes` et `Paquet` en suivant les descriptions précédentes.
3- Ecrire une **fonction** `genere_Carte` qui à partir d'une `valeur` et d'une `couleur` passés en paramètres remplit et retourne une `Carte` ayant ces caractéristiques.
4- Ecrire une **procédure** `affiche_Carte` affichant les informations relatives à une `Carte` (ex : `8 | CARREAU`)
5- Ecrire un sous-programme `genere_Paquet` qui génère **automatiquement** un `Paquet` de 52 `Cartes` triées (13 `Cartes PIQUE`, 13 `Cartes COEUR`, …). On pourra utiliser le sous-programme `genere_Carte`.
6- Ecrire une procédure `affiche_Paquet` qui affiche le `Paquet` généré ainsi que le nombre de `Cartes` qu'il comporte.
7- **Avant d'aller plus loin**, testez la génération et l'affichage du Paquet de 52 Cartes !

Le `Paquet` de `Cartes` ainsi obtenu est trié, nous allons maintenant le mélanger.

8- Ecrire une **procédure** `echange_Cartes` qui à partir d'un `Paquet` de `Cartes` et de 2 indices i et j échange les `Cartes` situées respectivement aux indices i et j dans le tableau `T` de `Cartes`.

9- En utilisant la procédure précédente, écrire un sous-programme `melange_Paquet` permettant de tirer aléatoirement 30 fois de suite les indices des `Cartes` à permuter dans le tableau de `Cartes` et en fait l'échange.
10- Affichez le `Paquet` ainsi obtenu pour vérifier vos deux derniers sous-programmes.

Nous allons maintenant distribuer les `Cartes` entre les 2 joueurs.

11- Ecrire une procédure `divise_Paquet` qui à partir d'un `Paquet` de 52 `Cartes` mélangées, créé et "retourne" 2 `Paquets` de 26 `Cartes` notés `jeuJ1` et `jeuJ2`. La distribution des `Cartes` se fera en alternance dans le jeu de chaque joueur.
12- Complétez votre programme principal pour affichez les 2 jeux obtenus.

BONUS

Une `partie` de bataille se limitera pour nous à un seul tour de jeu. Nous allons pour cela définir une structure `partie` comportant pour chaque joueur le `Paquet` de `Cartes` qu'il possède avant et après le tour de jeu. On pourra soit utiliser 2 `Paquets` différents pour chaque joueur (`J1_initial`, `J1_final`, `J2_initial`, et `J2_final`), soit faire un tableau de 2 `Paquets` pour chaque joueur `JeuJ1[2]` et `JeuJ2[2]` (dans la case 0 le `Paquet` initial, dans la case 1 le `Paquet` final).

13- Définir la structure `partie`.
14- Ecrire une procédure `bataille` qui va simuler un tour de jeu de bataille. La `partie` passée en paramètre contiendra initialement les `Paquets` des 2 joueurs issus de la distribution du `Paquet` de `Cartes` complet, et permettra de générer les `Paquets` de Cartes après comparaison de chacune des `Cartes` de leur `Paquet` initial.
On supposera ici que les 2 `Paquets` comportent le même nombre de `Cartes`.

15- Complétez le programme principal pour simuler un tour de jeu de bataille. Vous afficherez les `Paquets` finaux des 2 joueurs ainsi que leur score. Vous pourrez ensuite afficher qui remporte ce jeu.

```
Jeu initial
1 PIQUE || 2 PIQUE || 3 PIQUE || 4 PIQUE || 5 PIQUE || 6 PIQUE || 7 PIQUE || 8 PIQUE || 9 PIQUE || 10 PIQUE || 11 PIQUE || 12 PIQUE || 13 PIQUE ||
1 COEUR || 2 COEUR || 3 COEUR || 4 COEUR || 5 COEUR || 6 COEUR || 7 COEUR || 8 COEUR || 9 COEUR || 10 COEUR || 11 COEUR || 12 COEUR || 13 COEUR ||
1 CARREAU || 2 CARREAU || 3 CARREAU || 4 CARREAU || 5 CARREAU || 6 CARREAU || 7 CARREAU || 8 CARREAU || 9 CARREAU || 10 CARREAU || 11 CARREAU || 12 CARREAU || 13 CARREAU ||
1 TREFLE || 2 TREFLE || 3 TREFLE || 4 TREFLE || 5 TREFLE || 6 TREFLE || 7 TREFLE || 8 TREFLE || 9 TREFLE || 10 TREFLE || 11 TREFLE || 12 TREFLE || 13 TREFLE ||

Jeu melange
1 PIQUE || 9 CARREAU || 3 COEUR || 7 TREFLE || 6 COEUR || 6 PIQUE || 8 TREFLE || 8 PIQUE || 12 TREFLE || 7 COEUR || 10 PIQUE || 12 CARREAU ||
4 PIQUE || 11 COEUR || 9 TREFLE || 3 TREFLE || 4 COEUR || 11 CARREAU || 5 PIQUE || 2 TREFLE || 7 CARREAU || 9 COEUR || 10 COEUR || 8 COEUR ||
3 PIQUE || 13 COEUR || 1 CARREAU || 1 COEUR || 3 CARREAU || 4 CARREAU || 5 CARREAU || 6 CARREAU || 11 TREFLE || 12 COEUR || 2 PIQUE || 10 CARREAU ||
7 PIQUE || 8 CARREAU || 13 CARREAU || 1 TREFLE || 11 PIQUE || 12 PIQUE || 4 TREFLE || 6 TREFLE || 2 CARREAU || 13 PIQUE || 2 COEUR || 13 TREFLE ||
10 TREFLE || 5 TREFLE || 5 COEUR || 9 PIQUE ||

Paquet du joueur 1 : 26 cartes
1 PIQUE || 3 COEUR || 6 COEUR || 8 TREFLE || 12 TREFLE || 10 PIQUE || 4 PIQUE || 9 TREFLE || 4 COEUR || 5 PIQUE || 7 CARREAU ||
10 COEUR || 3 PIQUE || 1 CARREAU || 3 CARREAU || 5 CARREAU || 11 TREFLE || 2 PIQUE || 7 PIQUE || 13 CARREAU || 11 PIQUE ||
4 TREFLE || 2 CARREAU || 2 COEUR || 10 TREFLE || 5 COEUR ||

Paquet du joueur 2 : 26 cartes
9 CARREAU || 7 TREFLE || 6 PIQUE || 8 PIQUE || 7 COEUR || 12 CARREAU || 11 COEUR || 3 TREFLE || 11 CARREAU || 2 TREFLE ||
9 COEUR || 8 COEUR || 13 COEUR || 1 COEUR || 4 CARREAU || 6 CARREAU || 12 COEUR || 10 CARREAU || 8 CARREAU || 1 TREFLE ||
12 PIQUE || 6 TREFLE || 13 PIQUE || 13 TREFLE || 5 TREFLE || 9 PIQUE ||

APRES LA BATAILLE

Paquet du joueur 1 : 15 cartes
6 COEUR || 8 TREFLE || 12 TREFLE || 7 COEUR || 9 TREFLE || 3 TREFLE || 5 PIQUE || 2 TREFLE || 10 COEUR ||
8 COEUR || 1 CARREAU || 13 CARREAU || 1 TREFLE || 10 TREFLE || 5 TREFLE ||

Paquet du joueur 2 : 37 cartes
1 PIQUE || 9 CARREAU || 3 COEUR || 7 TREFLE || 6 PIQUE || 8 PIQUE || 10 PIQUE || 12 CARREAU || 4 PIQUE ||
11 COEUR || 4 COEUR || 11 CARREAU || 7 CARREAU || 9 COEUR || 3 PIQUE || 13 COEUR || 1 COEUR || 3 CARREAU ||
4 CARREAU || 5 CARREAU || 6 CARREAU || 11 TREFLE || 12 COEUR || 2 PIQUE || 10 CARREAU || 7 PIQUE || 8 CARREAU ||
11 PIQUE || 12 PIQUE || 4 TREFLE || 6 TREFLE || 2 CARREAU || 13 PIQUE || 2 COEUR || 13 TREFLE || 5 COEUR || 9 PIQUE ||

Le joueur 2 a gagne la partie

Process returned 0 (0x0)   execution time : 0.160 s
Press any key to continue.
```