POLYTECH SORBONNE

Sorbonne Université
2021-2022

Cours de Programmation en C – ROB 3

Travaux Pratiques

Objectif(s)

- ★ Définition de nouveaux types
- ★ Structures de données
- ★ Unions et enum
- ★ Passage par adresse

D'une manière simplifiée, On peut considérer une partie de black-jack de la manière suivante :

- la banque (l'ordinateur) et le joueur reçoivent chacun deux cartes, face visible
- Le principe est d'approcher le plus possible 21 points avec ses cartes, sans dépasser ce score. La valeur des cartes est la suivante :
  - de 2 à 10 : valeur nominale de la carte
  - chaque figure : 10 points
  - l'as : 1 ou 11 points, au choix

  Un "black-jack" est obtenu en faisant 21 points avec deux cartes (donc avec un as et un dix, valet, dame ou roi).
- Le joueur peut décider de demander autant de cartes supplémentaires qu'il veut. S'il dépasse 21 points, il perd.
- Quand le joueur ne veut plus de cartes supplémentaires, l'ordinateur peut demander des cartes supplémentaires.
- Celui qui a le plus de points, sans dépasser 21 points, a gagné la manche.

Dans un casino, ce jeu est bien entendu un jeu d'argent où l'on mise à chaque tour.
Le but de ce TP est de simuler le déroulement d'une partie. On commencera par gérer une carte, un jeu de carte, puis enfin une partie de carte.

1 Cartes

Question 1

Déclarer un nouveau type de donnée `t_carte` qui est une structure composée de

- un entier indiquant la valeur de la carte (1 pour as, 1 pour 2, ..., 11 pour valet, 12 pour dame et 13 pour roi)
- une valeur de type *énumération* indiquant la couleur (0 pour coeur, 1 pour pique, 2 pour carreau et 3 pour trèfle)

Question 2

Écrire une procédure acceptant une carte en paramètre d'entrée et qui affiche la carte :

- la procédure affichera un espace
- puis la valeur de la carte sur 2 caractères (`"␣A"` pour l'as, `"␣2"` pour le deux, ..., `"␣9"` pour le 9, `"10"` pour le 10, `"␣V"` pour le valet, `"␣D"` pour la dame et `"␣R"` pour le roi)
- puis la couleur:
  en UTF-8 (l'encodage de caractère utilisé sous votre session Linux), les couleurs s'obtiennent en affichant les chaines de caractères suivantes:
  - ♠ : `"\xE2\x99\xA0"`

- ♡ : "\xE2\x99\xA1"
- ♢ : "\xE2\x99\xA2"
- ♣ : "\xE2\x99\xA3"

- puis un dernier espace.

Au final, chaque carte est affichée sur 5 caractères. Par exemple, la fonction affichera "␣␣2♠␣", "␣␣A♡␣", "␣10♢␣" ou encore ␣␣R♣␣".

Pour tester cette fonction, on écrira un programme principal qui créé et affiche les quatre cartes suivantes : 2♠, A♡, 10♢ et R♣.

Question 3
Organiser le programme, pour que le type `t_carte` et le prototype de la procédure d'affichage se trouvent dans le fichier `carte.h`, la procédure d'affichage dans le fichier `carte.c` et le programme principal (de test pour le moment) dans un fichier `blackjack.c`.
Écrire le fichier `Makefile` permettant de compiler et faire l'édition de lien.

2 Jeu de cartes

Question 4
Déclarer un nouveau type `t_jeu` qui est un tableau de 52 cartes.
Écrire une fonction qui accepte un tableau de cartes en paramètre et qui le remplit (avec les 52 cartes possibles, de l'as au roi, de coeur au trèfle). L'ordre du remplissage importe peu.

Question 5
Écrire une procédure qui accepte un tableau de cartes en paramètre et l'affiche (affichage à la suite des 52 cartes).

Question 6
Pour mélanger les cartes, on se propose de permuter des cartes 2 à 2 un grand nombre de fois.
Écrire une procédure qui accepte un jeu de cartes et un entier (nombre de permutations) en paramètre, et qui permute autant de fois que donné en entrée deux cartes tirées au hasard.
Pour obtenir un numéro de carte au hasard, il faut utiliser la fonction `rand()` définie dans `stdlib.h`. Son prototype est: `long int random(void);`
Cette fonction renvoie un entier long aléatoire compris entre 0 et `RAND_MAX` (une constante qui vaut $2^{31}-1$ sur ce système). Pour obtenir un entier entre 0 et 51, il suffit d'appliquer une règle de trois. On peut aussi initialiser le générateur de nombre aléatoire avec la fonction `srand`. On utilisera cette fonction une seule fois dans le `main` avec l'instruction `srand( time(NULL));`, où la fonction `time` est définie dans `time.h` et renvoie l'heure courante.

Vérifier que cette procédure marche dans le programme principal :

- en créant un jeu de carte et en le remplissant
- en l'affichant
- en le mélangeant (par exemple avec 1000 fois)
- et en affichant le jeu mélangé (vérifiez s'il est bien mélangé)

3 Partie de black-jack

Question 7
On a besoin maintenant de modéliser la pioche, la main de l'ordinateur et la main du joueur.
La pioche est représentée par un jeu de cartes (qu'on peut appeler `pioche`) et la position de la prochaine carte à distribuer (qu'on appellera `posPioche`).
La main d'un joueur (joueur humain ou ordinateur) sera aussi représentées par un jeu de cartes (qu'on appellera `jeu` par exemple) ,le nombre de cartes qu'elle contient (`nbCartes` par exemple), ainsi qu'un tableau de 20 caractères pour y stocker le nom du joueur qui possède la main.
Les cartes d'indice `0` jusqu'à l'indice `nbCartes` du tableau `jeu` seront donc les cartes de la main du joueur.

Dans un fichier `partie.h`, créer un type structuré pour modéliser une main. Créer un type structuré `t_partie` pour modéliser une partie (pioche et deux mains).

Question 8

Dans `partie.c`, écrire une procédure qui initialise une partie passée en paramètre. Deux chaînes de caractères (des tableaux de 20 caractères) sont aussi passés en paramètres, pour indiquer le nom des deux joueurs à qui appartiennent les mains
Pour cela, il faut donc

- remplir et mélanger la pioche
- indiquer qu'elle commence à la carte n° 0
- les mains doivent n'avoir aucune carte
- recopier les noms passés en paramètres dans les noms des mains. Pour cela, il faut utiliser la procédure `strcpy` (définie dans `string.h`) qui prend en 1er argument le tableau de caractère où copier (la destination), et en 2nd argument le tableau de caractères à copier (la source).

Attention, vérifiez bien quels paramètres sont modifiés par la fonction, et s'il faut faire un passage par valeur ou par adresse.

Question 9

Distribuer une carte de la pioche vers la main du joueur (ou de l'ordinateur) revient donc à copier la carte d'indice `posPioche` du tableau `pioche` dans la carte d'indice `nbcJoueur` du tableau `jeuJoueur`, et à incrémenter ces deux valeurs (idem pour la distribution à l'ordinateur).
Écrire, dans un fichier `partie.c` une procédure qui accepte 2 arguments :

- une partie
- un entier indiquant à quel joueur il faut distribuer (`0` pour l'ordinateur, `1` pour le joueur humain)

et qui distribue une carte de la pioche dans la main.

Question 10

Écrire une procédure qui affiche les cartes contenues dans une main (cette procédure prendra donc en paramètres une main).

Pour tester ces deux procédures, écrire un programme principal qui

- créé une partie, avec aucune carte distribuée dans les mains (et donc une pioche commençant à l'indice 0) ;
- indique le nom des deux joueurs ;
- remplit puis mélange la pioche ;
- affiche la pioche (pour pouvoir vérifier ensuite que la distribution s'est bien déroulée) ;
- distribue une carte au joueur, une carte à l'ordinateur puis encore une carte au joueur et une carte à l'ordinateur ;
- affiche la main des deux joueurs (précédé de leur nom).

Question 11

Écrire une fonction qui calcule la meilleure valeur d'une main donnée. Il s'agit donc d'additionner les valeur de toutes les cartes d'une main.
La seule difficulté concerne les as, qui peuvent valoir 1 ou 11 points, au choix. Pour cela, on considère d'abord que les As valent 1 point uniquement.
Ensuite, si la main vaut moins de 11 points et si on a au moins un as, alors il faut compter un as comme 11 points (cela revient à rajouter 10 points, tout simplement). Ça ne sert à rien de compter d'autres as comme valant 11 points, car sinon, on dépasse forcément les 21 points.
Tester cette fonction sur les deux mains (ordinateur et joueur).

Question 12

Il faut maintenant distribuer au joueur autant de cartes qu'il désire.
On propose l'algorithme suivant :

```
Distribution d'une carte pour l'ordinateur
Distribution d'une carte pour le joueur
Distribution d'une carte pour l'ordinateur
Distribution d'une carte pour le joueur
Affichage des deux mains
Demander si le joueur veut une nouvelle carte
tant que reponse=oui et valeur du jeu ≤ 21 faire
    distribution d'une carte pour le joueur
    affichage de la main du joueur
    si valeur du jeu ≤ 21 alors
        demander si le joueur veut une nouvelle carte
    fin si
fin tantque
```

Écrire le programme correspondant.
Remarque : pour lire la réponse, il est préférable de lire un tableau de caractères (avec `"%s"`), afin de s'affranchir du problème du caractère de fin de ligne que l'on récupère aussi lorsque l'on utilise `"%c"` dans le scanf.

4 Bonus

Question 13

Enfin, maintenant, il faut écrire la "stratégie" de l'ordinateur. En réalité, dans les casino, le croupier joue avec une règle simple résumée par "La banque tire à 16, reste à 17". Ainsi, tant que le jeu de l'ordinateur est inférieur à 16, il reprend une carte. Dès qu'il dépasse 17 (inclus), il s'arrête.
Écrire la stratégie de jeu de l'ordinateur, à la suite du programme précédent. Indiquer le vainqueur.
Rajouter une boucle pour jouer tant qu'il reste au moins 10 cartes dans la pioche.