Université Polytechnique Hauts-de-France
Institut National des Sciences Appliquées Hauts-de-France **2023-2024**
FISE 3 - MT Langage C
Semestre 5

Fiche d'exercices pratiques 2

Tous les algorithmes doivent être écrits en langage `C`.

Exercice 1

1. Écrire une fonction `int plus_moins(int n)` qui simule le jeu du "plus ou moins". La valeur de `n` correspond au nombre à deviner. Cette valeur est *choisie par l'ordinateur* aléatoirement, entre 1 et 999 (en utilisant la fonction `rand`). L'utilisateur cherche à deviner cette valeur. La fonction retourne le nombre de coups nécessaires pour trouver le nombre.

Après avoir implémenté la fonction vous la tester via une fonction `main`.

2. Écrire ensuite une fonction `int plus_moins_ordi(int n)` qui permet de simuler une partie dans laquelle c'est l'ordinateur qui cherche la valeur `n` donnée. On implémentera ici une version "sans intelligence" dans laquelle l'ordinateur propose une valeur aléatoire à chaque tentative, mais dans l'intervalle possible des valeurs. Il pourrait par exemple proposer en premier 247. Si l'utilisateur répond "plus" un nombre est tiré au hasard entre 248 et 999, et ainsi de suite. La fonction retourne le nombre de coups nécessaires pour trouver le nombre.

Remarque : pour simplifier le processus il est conseillé d'utiliser la saisie d'un entier pour simuler la réponse à la question "plus ou moins ?", plutôt qu'un caractère.

3. Écrire enfin une fonction `int plus_moins_ordi_mieux(int n)` dans laquelle on implémente une version cherchant à minimiser le nombre de coups (en moyenne), ce qui est réalisable en découpant l'intervalle de recherche par deux à chaque itérations. Ici encore la fonction retournera le nombre de coups nécessaires pour trouver la valeur choisie par l'utilisateur.

Exercice 2

On s'intéresse à une version simple du jeu du "Mastermind" qui consiste à trouver une combinaison de 4 couleurs choisies parmi 8 en un maximum de 10 tentatives. Pour simplifier on utilisera ici les entiers de 1 à 8 pour représenter les 8 couleurs. Le principe du jeu est le suivant : l'ordinateur choisit une combinaison de 4 "couleurs" (4 entiers entre 1 et 8 ici) aléatoirement, sachant que dans cette version simplifiée une couleur ne peut être utilisée qu'une seule fois au plus. Cette combinaison est stockée dans un tableau. Le joueur va ensuite proposer une première combinaison (stockée dans un autre tableau). L'algorithme donne des indications au joueur en fonction de la proposition et de la combinaison cherchée : il affiche un 'N' (assimilé au pion noir dans le jeu) pour chaque "couleur" correctement placée dans la proposition, et un 'B' (assimilé au pion blanc dans le jeu) pour chaque "couleur" présente dans la combinaison mais pas correctement placée. Le joueur peut alors faire une nouvelle proposition, et le processus s'arrête dès que la combinaison est trouvée ou que 10 propositions ont été vérifiées.

1. Définir type `TABC` pour stocker les combinaisons.
2. Écrire une fonction `void initialise(short t[4])` de génération de la combinaison par l'ordinateur. Attention au fait que chaque couleur (chaque entier) ne peut être utilisée qu'une seule fois.
3. Écrire une fonction `void recup_coup(short t[4])` de saisie d'une combinaison par le joueur. Cette fonction doit s'assurer que la tentative est correcte (valeurs distinctes et entre 1 et 8, on interdit au joueur de proposer une combinaison avec une même couleur présente plusieurs fois).
4. Écrire une fonction `void affiche_coup(short coup[4])` qui affiche une proposition à l'écran.

5. Écrire une fonction `int verif_coup(short combinaison[4], short coup[4])` de vérification d'une proposition. Cette fonction affiche les "pions" blancs et noirs à l'écran.
6. Écrire une fonction `void mastermind()` de gestion du jeu (initialisation, répétition des tours, ...).
7. Tester le tout avec une fonction `main`.

Proposer une autre version dans laquelle c'est le joueur qui choisit la combinaison, et l'ordinateur qui la cherche (de façon "non intelligente").

[Bonus] Proposer une autre version permettant de traiter le cas où une même couleur peut être présente plusieurs fois dans la combinaison.