Exercice fonction

Exercice 1 loop

Écrire une fonction `MaxValueInTab`, qui va parcourir un tableau d'entier et qui va retourner la valeur maximale du tableau.

L'objectif étant de parcourir le tableau avec une boucle `for` et de non d'utiliser une méthode.

Exercice 2 Loop

Écrire une fonction `PrintTabInSortWay` qui prend en paramètre un tableau d'entier et qui affiche le tableau dans l'ordre croissant.

L'objectif est de ne pas trier le tableau ni de faire de copie du tableau. Mais de le parcourir et de l'afficher dans l'ordre croissant.

Exercice 3

- 📌 Écrire `GetRandomNumber(int min, int max)` : Cette fonction prend en paramètres deux entiers min et max et retourne un nombre entier aléatoire compris entre min et max (inclus).
- 📌 Écrire `GetRandomAttack()` : Cette fonction retourne un nom d'attaque aléatoire sous forme de chaîne de caractères. Les attaques possibles sont "coup de poing", "coup de pied", "projection" et "étranglement".

📌 Écrire `GetAttackPower(string attack)` : Cette fonction prend en paramètre un nom d'attaque (attack) et retourne la puissance de l'attaque sous forme d'entier. La puissance des attaques est la suivante: "coup de poing" = 5, "coup de pied" = 10, "projection" = 15, "étranglement" = 20.

📌 Écrire

```
Attack(ref int player1Health, ref int player2Health, string attack)
```

: Cette fonction prend en paramètres les points de vie de deux joueurs (player1Health et player2Health) et un nom d'attaque (attack) et met à jour les points de vie des joueurs en fonction de l'attaque.
Si l'attaque est "coup de poing" ou "coup de pied", le joueur 1 attaque le joueur 2 et réduit ses points de vie de la puissance de l'attaque. Si l'attaque est "projection" ou "étranglement", le joueur 2 attaque le joueur 1 et réduit ses points de vie de la puissance de l'attaque.
La fonction ne retourne rien mais affiche le nom de l'attaque et les points de vie des joueurs après l'attaque.

Exercice 4

Voici les fonctions que vous devrez écrire pour ce programme:

`RollDice` : cette fonction prend en paramètre un entier `sides` indiquant le nombre de faces du dés et retourne un nombre aléatoire compris entre 1 et sides.
Vous devrez utiliser la classe Random pour générer le nombre aléatoire.

Exercice 4

`RollDiceMultipleTimes` : cette fonction prend en paramètre un entier `sides` indiquant le nombre de faces du dés et un entier `count` indiquant le nombre de dés à lancer. La fonction appelle la fonction `RollDice` pour lancer chaque dé et retourne un tableau à une dimension contenant les résultats de chaque lancer.

`GetTotalScore` : cette fonction prend en paramètre un tableau à une dimension `results` contenant les résultats de lancers de dés et retourne le score total en additionnant tous les éléments du tableau.

Exercice 4

`PrintDiceRollResults` : cette fonction prend en paramètre un tableau à une dimension `results` contenant les résultats de lancers de dés et affiche le résultat de chaque lancer sur la console.

`PlayDiceGame` : cette fonction utilise les quatre fonctions précédentes pour simuler un lancement de dés. La fonction demande à l'utilisateur combien de dés il souhaite lancer, appelle la fonction RollDiceMultipleTimes pour lancer les dés, appelle la fonction GetTotalScore pour calculer le score total et appelle enfin la fonction PrintDiceRollResults pour afficher les résultats sur la console.

Exercice 5

Écrire une fonction

```
GetDaysBetweenTwoDates(int mouth1, int day1, int year1, int mouth2, int day2, int year2)
```

qui prend en paramètres deux dates et qui retourne le nombre de jours qui sépare les deux dates.

Exercice 5 exemple

```
GetDaysBetweenDates(12, 31, 2020, 1, 1, 2021) => 1
GetDaysBetweenDates(2, 28, 2020, 3, 2, 2020) => 2
```