Travaux Pratiques 2

Objectifs:

- Utiliser les instructions de sélection (if, else)
- Utiliser les instructions d'itération (for, while)
- Manipuler les enums et les structs
- Manipuler les tableaux
    - Unidimensionnel
    - Multidimensionnel
    - En escaler
- Utiliser des membres de classe défini dans un autre fichier
- Afficher un résultat dans la console

Critères de performances:

- 3.1 Résolution efficace des problèmes de conception de l'algorithme.
- 3.2 Résolution efficace des problèmes de traduction de l'algorithme dans le langage de programmation.
- 3.3 Résolution efficace des problèmes d'utilisation du langage de programmation.
- 4.3 Interprétation juste des résultats.
- 4.4 Fonctionnement correct du programme.

Date de remise: 14 février 2018

Instructions:

1. Téléchargez le projet Visual Studio ici: https://github.com/felixsoum/420JV4AS-TP2
2. Écrivez votre nom en commentaire au début du fichier Program.cs
3. Faites tout le travail dans Program.cs et soumettre seulement ce fichier sur Léa

Chaque question correspond à une fonction. Les fonctions doivent être implémenté correctement pour que l'appel de ces fonctions affichent des résultats qui correspondent à ceux dans ce document.

Question 1. Tri des minéraux

Il faut souvent trier des éléments pour faciliter leur analyse ou simplement les afficher en ordre pour l'utilisateur. Le triage d'éléments est un sujet de recherche important et cette question vous donnera un aperçu.

Implémentez une fonction pour trier des enums qui représentes des minéraux. Il faut les trier en ordre croissant de qualité. La fonction contient déjà du code pour afficher le tableau, donc il ne faut que le trier.

Le paramètre d'entré de la fonction est un tableau unidimensionnel contenant les enums. La classe Data fournit les données pour cette question.

Résultats:

```
// Question1(Data.Ores1)
Copper, Silver, Gold,
```

```
// Question1(Data.Ores2)
Copper, Copper, Copper, Silver, Silver, Gold,
```

Question 2. Analyse des tuiles 2D

Les tuiles dans les jeux 2D sont souvent représentées par des tableaux à deux dimensions. Il est alors important de se familiariser avec l'itération dans un tableau multidimensionnel pour analyser les données.

Implémentez une fonction pour trouver la meilleur tuile qui contient le minerai spécifié avec la plus grande valeur parmi d'autres. La valeur d'une tuile se calcule par la somme de la valeur du minerai sur la tuile et de tous les minéraux du même type dans les tuiles adjacentes (nord, est, sud, ouest). Après avoir trouvé la meilleur tuile, affichez ses index [x, y] du tableau.

Le premier paramètre la fonction est le type de minerai à rechercher et le deuxième est le tableau des tuiles. La classe Data fournit les données pour cette question.

Résultats:

```
// Question2(Ore.Silver, Data.Tiles)
1, 1
```

```
// Question2(Ore.Gold, Data.Tiles)
3, 2
```

Question 3. Affichage en escalier

Implémentez une fonction qui affiche les tableaux (à l'intérieur du tableau en escalier) en ordre décroissant de taille. Si plusieurs tableaux ont la même taille, alors le tableau avec le plus grand total des éléments (la somme numérique) prends priorité.

Le paramètre d'entré de la fonction est un tableau en escalier contenant des tableaux de nombre entier. La classe Data fournit les données pour cette question.

Résultats:

```
// Question3(Data.Jagged1)
0, 2, 4, 6, 8,
1, 3, 5, 7,
10, 12,
```

```
// Question3(Data.Jagged2)
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
0, 2, 4, 6, 8, 0, 2, 4, 6,
0, 1, 2, 3, 4, 5, 6, 7,
0, 1, 2, 3, 3, 5, 6, 7,
0, 1, 0, 1, 0, 1, 0, 1,
8, 6, 4, 2, 2, 4, 6,
7, 6, 5, 4, 3, 2,
2, 1, 2, 1, 2,
1, 2, 1, 2, 1,
2, 3, 5, 7,
8, 8, 8,
7, 7, 7,
0, 1,
1,
```