Formatif 1 – C# (3 semaines)

Quelques consignes :

- Créez une solution Application Console C# Formatif_1.
- Tous les programmes demandés sont des projets différents de la solution Formatif_1.
- Dans tous les cas où c'est pertinent, gérez les exceptions.
- Dans tous les cas, créez des méthodes non statiques pour bien diviser votre code.
- Exploiter au mieux les possibilités du langage C#.

Environnement

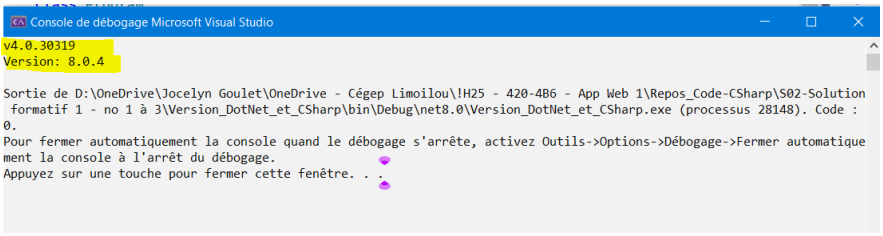
A. Commencez par installer Visual Studio Enterprise sur vos équipements personnels et vérifier avec un premier programme « Console » que tout fonctionne bien.

Par programmation :

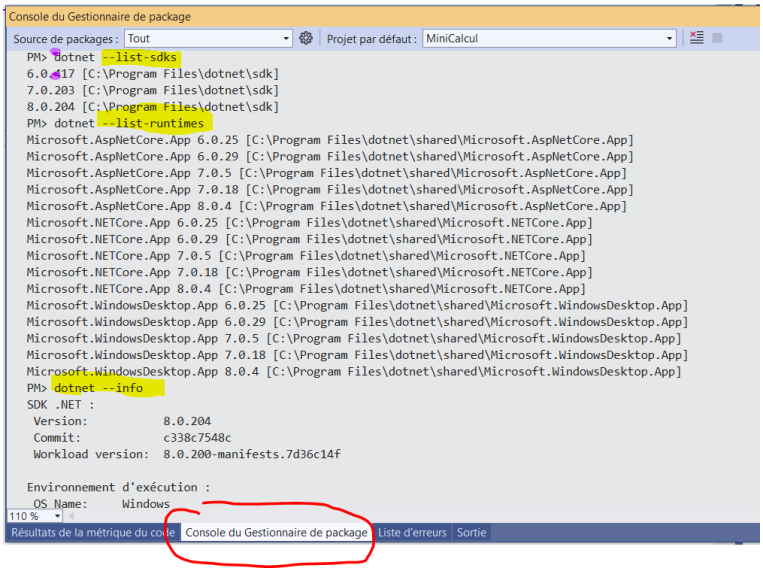
```
//Quelle version de .NET j'utilise
Console.WriteLine(typeof(string).Assembly.ImageRuntimeVersion);

//Quelle version du langage C# - ça devrait être 8 et plus
Console.WriteLine("Version: {0}", Environment.Version.ToString());
```

Sortie par programmation :



En mode console : https://docs.microsoft.com/fr-ca/dotnet/core/install/how-to-detect-installed-versions?pivots=os-windows



Si vous n'avez pas la version .NET 8, modifier le csproject (double-clic sur projet) et assurez-vous que le targerFramework est net 8.0. Revérifier, si ça ne fonctionne toujours pas, allez télécharger .NET 8 https://dotnet.microsoft.com/en-us/download/dotnet.

B. Bâtissez une équipe de travail (2 ou 3 personnes) et remplissez le formulaire d'équipe, disponible dans le dossier du cours, semaine 1.

Projets à développer (complexité croissante)

1. Créer un projet console **Convertisseur** qui permet de convertir des degrés Celsius en Kelvin et en Fahrenheit. Demandez la **saisie** à l'utilisateur et permettez-lui de saisir de nouveau s'il y a une erreur et de refaire une conversion s'il le désire. Utilisez une classe pour diviser votre code en méthodes (pas de méthodes statiques). Construisez un « *main »* pour tester votre application.

2. Créer un projet console **MiniCalcul** offrant les opérations élémentaires suivantes « +, -, /, * ». Les opérations disponibles font partie d'une **énumération** et c'est ainsi que l'on pourra valider le choix de l'opérateur.

   L'utilisateur saisit son opérateur, s'il est valide on lui demande deux valeurs, si elles sont valides on fait l'opération et on affiche le résultat.

   Gérer les **exceptions** et redemander la saisie si l'utilisateur commet une erreur.

3. Créer un projet **Palindrome** qui permet de détecter si la chaîne saisie par l'utilisateur est un palindrome (mot qui peut être lu dans les deux sens). Si c'est le cas, reproduisez le palindrome dans un motif en utilisant une boucle **foreach** (lettres sur des lignes séparées et progressives et alternance majuscules et minuscules), par exemple :

   K
   Ka
   KaY
   KaYa
   KaYaK

   Demandez à l'utilisateur de saisir un mot, gérez les exceptions et redemandez si l'utilisateur veut faire un autre essai. **Nettoyez** l'écran avant d'afficher le nouveau palindrome. Dans le cas où ce n'est pas un palindrome, on affiche que la chaîne qui ne constitue pas un palindrome.

   Faites un traitement itératif ou **récursif** (si vous pensez être capable ;o)) pour vérifier si c'est un palindrome.

4. Créer un projet **LesRegex** qui demande à l'utilisateur une chaîne de caractères et affiche les statistiques de cette chaîne : nombre de majuscules avec un Regex, nombre de signes de ponctuation avec un Regex, nombre de caractères …

5. Créer un projet **Vecteur** qui génère un tableau dynamique de 20 entiers aléatoirement (il peut y avoir des doublons) dont les valeurs sont entre 33 et 127. Avec ce tableau, effectuez les opérations suivantes :

   a) Afficher le tableau généré selon l'ordre de création.

   b) Afficher le tableau **trié** en ordre croissant sans modifier le tableau initial.

   c) Afficher uniquement les valeurs impaires triées en ordre décroissant.

   d) Afficher la valeur maximale de ce tableau.

   e) Afficher les valeurs du tableau qui sont supérieures à la valeur saisie par l'utilisateur (faire une saisie).

   f) Retourner un nouveau tableau statique contenant le caractère **alphanumérique** (valeur ascii) de la valeur numérique de votre tableau si celle-ci existe et afficher ce nouveau tableau (exemple : j'ai **33, 56, 74** ça me retournerait : « **!, 8, J** »).

   Une de vos méthodes doit utiliser la boucle **foreach**. Vous devez redéfinir la méthode **ToString** pour l'affichage du tableau primaire. Le tout doit être présenté clairement à l'écran.

6. Créer un projet **Structure** qui permet de conserver les valeurs « rvb » (rouge, vert, bleu) d'une couleur et son nom dans une structure **« struct »**. Par la suite, effectuez les traitements demandés à partir de cette structure.

   a. Créer une couleur et l'enregistrer dans un vecteur (peupler votre vecteur avec 4 ou 5 couleurs).

   b. Afficher l'ensemble des couleurs du vecteur.

   c. Chercher une couleur par son nom et afficher son code « rvb ». (Chercher avec une couleur existante et inexistante).

   d. Effacer une couleur.

   e. Modifier le nom d'une couleur.

   Pour la validation, on ne fait que valider si le code de couleur primaire est entre 0 et 255, si ce n'est pas le cas on met 0 par défaut.

   Créer un petit « *main* » pour tester l'ensemble de vos traitements.

   Toujours dans votre « main » faites l'expérience suivante :

   f. Créez 2 couleurs c1 et c2 l'une avec ces valeurs : 128, 128, 128, "gris" et l'autre avec 255, 255, 255, "blanc".

   g. Maintenant, copiez la structure c2 dans c1 et constatez le résultat.

   h. Modifiez la couleur de c1 pour "rose" puis faites afficher les valeurs des structures c1 et c2. **Qu'est-ce que vous en concluez ?**

7. Créer un projet **Produits**, qui contient une classe **Produit** et des **tests unitaires** pour valider chacune des méthodes.

   Les attributs sont : **noProduit, nomProduit, description, prix, quantité, qteRupture, taxable.**

   L'attribut qteRupture contient le nombre minimal de la quantité en stock toléré.

   Le constructeur nécessite noProduit, nomProduit. Les autres attributs sont à 0 ou à une chaîne vide ou à "false" et sont complétés grâce aux "setter".

   Tous les attributs ont un getter/setter. Le prix doit être entre 0 et 500, la quantité doit être entre 0 et 200, la qteRupture doit être entre 0% et 20% de la quantité.

   On redéfinit les opérateurs d'égalité en lien avec le noProduit et les opérateurs « < » et « > » en lien avec la (quantité en stock - qté de rupture).

   Une méthode « Total » calcule la **valeur en stock** d'un produit (Quantite * Prix)

   Un ToString permet de retourner les attributs de l'objet, si plus grande que 0 pour les numériques et **la valeur en stock**.

   Créer un projet ProduitTest pour créer un test unitaire NUnit qui sert à vérifier chacune des méthodes publiques y compris les opérateurs de votre classe **Produit**. Assurez-vous d'avoir une couverture de test au-delà de 90%.