Préambule :

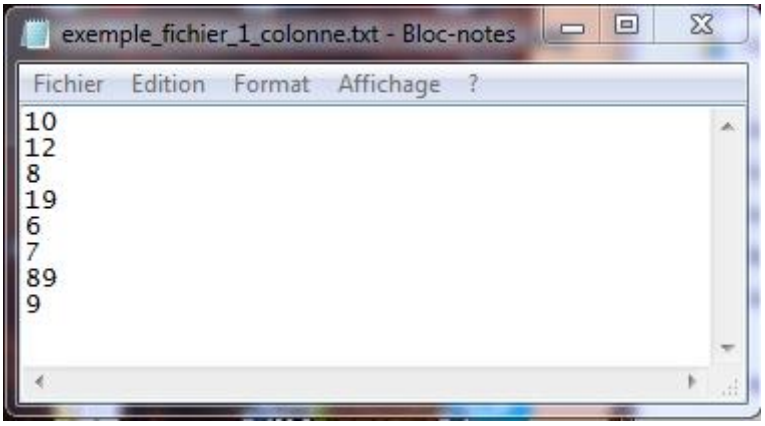
A. Chaque exercice correspond à un projet Lazarus.
B. Attention, le programme principal doit être le plus simple possible. Vous devez implémenter les fonctionnalités sous forme de procédures et fonctions.
C. Ces dernières doivent être obligatoirement dispatchées dans des unités ! Vous pouvez (à mon avis, c'est un bon conseil) réutiliser (importer) les corrections (les unités) des exercices précédents dans vos projets.
D. Plusieurs projets peuvent partager la même unité.

1. Exercice 1 – Fichier texte

Créer une application qui :

1. Demande à l'utilisateur le nombre d'éléments du tableau ;
2. Procède à la saisie des valeurs ;
3. Affiche les valeurs ;
4. Demande à l'utilisateur un nom de fichier (extension « .txt »);
5. Sauvegarde le contenu du tableau dans le fichier texte.

Voici un exemple de fichier sauvegardé, édité par la suite dans le bloc-notes de Windows



2. Exercice 2 – Fichier texte

Créer une application qui :

1. Demande à l'utilisateur un nom de fichier texte censé contenir une liste de valeurs numériques ;
2. Charge les valeurs dans le fichier dans un tableau ;

3. Affiche les valeurs du tableau.

Remarque 1 : **SetLength()** permet de changer dynamiquement la taille d'un tableau. Vous pouvez donc commencer avec un tableau de taille 1, puis de taille 2, etc. jusqu'à la fin du fichier.

Remarque 2 : Les fonctions **FloatToStr()** et **StrToFloat()** permettent d'effectuer la conversion d'un numérique en chaîne, et inversement. Elles sont implémentées dans l'unité SysUtils, vous devez donc ajouter la clause « ***uses*** *SysUtils ;* » dans votre unité.

3. Exercice 3 – Fichier typé

Créer une application qui permet de gérer une collection de véhicules avec les caractéristiques suivantes :

```
TVoiture = record
      marque : shortstring;
      age : integer;
      cv : integer;
    end;
```

Elle doit :

1. Demander à l'utilisateur le nombre de véhicules à gérer ;
2. Procéder à la saisie de leurs caractéristiques ;
3. Demander à l'utilisateur un nom de fichier de sauvegarde ;
4. Stocke les véhicules dans le fichier.

4. Exercice 4 – Fichier typé

Créer une application qui :

1. Demande à l'utilisateur un nom de fichier contenant des véhicules (TVoiture) ;
2. Charge les véhicules dans un tableau ;
3. Les trie selon la taxe qui leur est associée ;
4. Affiche à l'écran les caractéristiques des véhicules triées selon leur taxe.

5. Exercice 5 – Fichier texte (CSV)

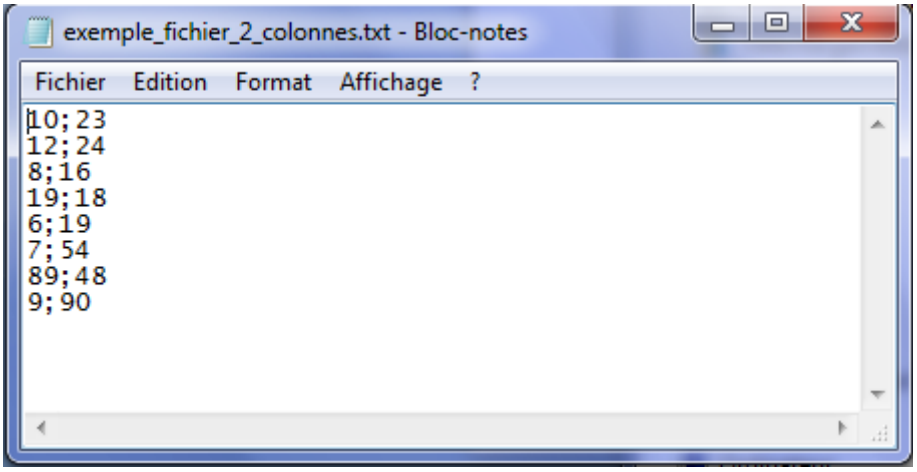
Créer une application qui :

1. Demande à l'utilisateur le nombre d'éléments ;
2. Réalise la saisie de 2 vecteurs de valeurs X et Y ;
3. Calcule et affiche la somme du produit croisé entre ces deux vecteurs

$$S = \Sigma_i X_i * Y_i ;$$

4. Demande à l'utilisateur un nom de fichier ;
5. Sauvegarde les 2 vecteurs dans ce seul et même fichier (les colonnes seront séparées par le caractère « ; »).

Voici un exemple de fichier ouvert dans le bloc-notes de Windows.



Remarque : L'opérateur **+** permet de réaliser la concaténation de 2 chaînes de caractères.

6. Exercice 6 – Fichier texte (CSV)

Créer une application qui :

1. Demande à l'utilisateur un nom de fichier contenant 2 colonnes de valeurs séparées par « ; » ;
2. Charge le contenu du fichier dans 2 vecteurs ;
3. Affiche le contenu des deux vecteurs.

Remarque : **pos()** permet de chercher une sous-chaîne dans une chaîne de caractères, **copy()** permet de copier une partie d'une chaîne de caractères, **length()** fournit la longueur d'une chaîne de caractères, **delete()** permet la suppression d'une partie d'une chaîne de caractères (cf. le fichier d'aide de Delphi).