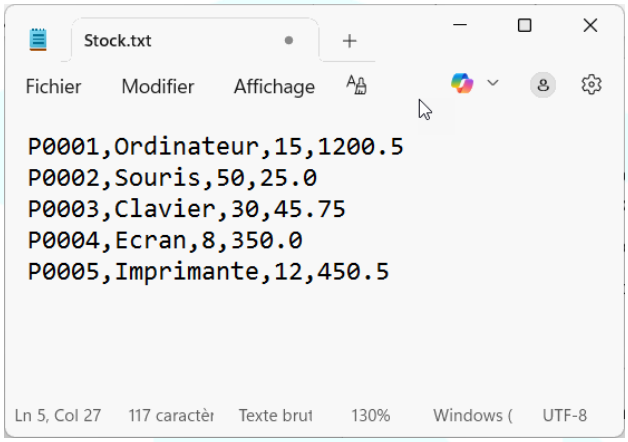
Travaux dirigés n°2 : Algorithmes de manipulation de Fichiers

Matière : Algorithmique et programmation
Sujet : Manipulations de base sur les fichiers

Une entreprise de vente de matériel informatique souhaite gérer son stock de produits à l'aide d'un fichier texte nommé **"Stock.txt"**.
Chaque ligne du fichier contient les informations d'un produit séparées par le caractère **","** (virgule) :

- **Code** : Code du produit (chaîne de caractères)
- **Nom** : Désignation du produit (chaîne de caractères)
- **Qte** : Quantité en stock (entier naturel strictement positif)
- **Pu** : Prix unitaire en dinars (réel strictement positif)

Exemple de contenu du fichier "Stock.txt" :



Instructions Générales :

- Les modules sont conçus pour être autonomes, chacun traitant une tâche spécifique de manière indépendante.
- Vous n'êtes pas appelé à saisir les données déjà fournies en paramètres. Tous les paramètres des procédures et fonctions sont considérés comme déjà validés et disponibles au moment de l'appel.
- Le nom physique du fichier est passé en paramètre (ex: NomF). Le nom logique (variable fichier) est déclaré localement.

Questions :

1. Écrire l'algorithme d'une **procédure Remplir_version1(nomf : Chaîne de caractères, n : Entier)** qui permet de créer et remplir le fichier "**Stock.txt**" avec **n** produits saisis au clavier.
   Remarque :
   - Chaque produit sera enregistré dans une ligne du fichier au format CSV (champs séparés par des virgules).
   - **n** est déjà saisi au niveau de programme appelant.
2. Écrire l'algorithme d'une **procédure Remplir_Version2(nomf : Chaîne de caractères)** qui permet de créer et de remplir le fichier texte "**Stock.txt**" par des produits saisis par l'utilisateur, jusqu'à ce que celui-ci réponde "**N**" à la question : « **Voulez-vous ajouter un produit (O/N) ?** »
3. Écrire l'algorithme d'une **fonction Compter_Produits(nomf : Chaîne de caractères) : Entier** qui permet de retourner le nombre de produits (lignes) contenus dans le fichier.

   Exemple : Si le fichier contient 5 lignes, la fonction retourne 5.

4. Écrire l'algorithme d'une **procédure Ajouter_Fin(nomf, code, nom : Chaîne de caractères, qte : Entier, pu : Réel)** qui permet d'ajouter un nouveau produit à la fin du fichier sans écraser le contenu existant.
   Remarque : Le prix unitaire et la quantité sont déjà validés au niveau du programme appelant.

5. Écrire l'algorithme d'une **procédure Ajouter_Debut(nomf, code, nom : Chaîne de caractères, qte : Entier, pu : Réel)** qui permet d'ajouter un nouveau produit au début du fichier (il devient à la première ligne).

6. Écrire l'algorithme d'une **fonction Rechercher(nomf, code_rech : Chaîne de caractères) : Booléen** qui permet de vérifier si un produit identifié par son code existe dans le fichier.
   Exemple : Rechercher(nomf, "P0001") retourne Vrai si le produit existe

7. Écrire l'algorithme d'une **procédure Ajouter_Si_Nouveau(nomf, code, nom : Chaîne de caractères, qte : Entier, pu : Réel)** qui permet d'ajouter un produit au fichier uniquement s'il n'existe pas déjà (vérification par code).
   Remarques :
   - Si le produit existe, afficher "Produit déjà existant"
   - Sinon, l'ajouter à la fin du fichier et afficher "Produit ajouté avec succès"
   - Utiliser la fonction Rechercher.

8. Écrire l'algorithme d'une procédure **Generer_Commande(nomf_source, nomf_commande : Chaîne de caractères, seuil : Entier)** qui permet de créer un fichier de commande contenant tous les produits dont la quantité est inférieure à un seuil donné.

   Format du fichier de commande : Code,Nom,Quantité à commander
   Exemple : Si seuil = 10 et le stock contient :

   ```
   P0001,Ordinateur,15,1200.50
   P0003,Clavier,5,45.75
   P0004,Écran,8,350.00
   ```

   Le fichier commande contiendra :

   ```
   P0003,Clavier,5
   P0004,Écran,2
   ```

   Remarque : La quantité à commander = seuil - quantité actuelle

9. Écrire l'algorithme d'une **procédure Filtrer(nomf, nomf_chers, nomf_eco : Chaîne de caractères, seuil : Réel)** qui permet de séparer les produits en deux fichiers selon leur prix :
   a. Les produits avec prix > seuil dans le fichier dont son nom physique est représenté par le paramètre "**nomf_chers**"
   b. Les produits avec prix ≤ seuil dans le fichier dont son nom physique est représenté par le paramètre "**nomf_eco**"

   Remarque : Les deux fichiers destination seront créés.

10. Écrire l'algorithme d'une procédure **Supprimer_Produit(nomf, code_supp : Chaîne de caractères)** qui permet de supprimer du fichier texte "**Stock.txt**" le produit dont le code est égal à code_supp. Si le produit n'existe pas, afficher un message.

11. Écrire l'algorithme d'une **fonction Valeur_Stock_Total(nomf : Chaîne de caractères) : Réel** qui permet de calculer et retourner la valeur totale du stock.
    Formule : Valeur totale = la somme de (quantité × prix unitaire) pour tous les produits
    Exemple :
    - P0001 : 15 × 1200.5 = 18007.5
    - P0002 : 50 × 25.0 = 1250.0
    - Valeur totale = 18007.5 + 1250.0 + ...

12. Écrire l'algorithme d'une **procédure Modifier_Prix(nomf, code_modif : Chaîne de caractères, nouveau_prix : Réel)** qui permet de modifier le prix unitaire d'un produit identifié par son code.
Remarques :
    - Le nouveau prix est déjà validé (> 0) au niveau du programme appelant.
    - Si le produit n'existe pas, afficher un message.

13. Écrire l'algorithme d'une **procédure Augmenter_Prix(nomf : Chaîne de caractères, pourcentage : Réel)** qui permet d'augmenter le prix de tous les produits du fichier d'un pourcentage donné.

    Exemple : Pour pourcentage = 10, tous les prix sont augmentés de 10%.
    - Si un produit coûte 100.0 DT, son nouveau prix sera 110.0 DT
    - Formule : nouveau_prix ← ancien_prix × (1 + pourcentage/100)

    Remarque : Le pourcentage est déjà validé (> 0) au niveau du programme appelant.

14. Écrire l'algorithme d'une **procédure Trier_Croissant(nomf : Chaîne de caractères)** qui permet de trier les produits du fichier par prix unitaire croissant (du moins cher au plus cher).

15. Écrire l'algorithme d'une **procédure Inserer_Trie(nomf, code, nom : Chaîne de caractères, qte : Entier, pu : Réel)** qui permet d'insérer un nouveau produit dans un fichier **déjà trié par prix croissant** tout en maintenant l'ordre du tri.

    Exemple : Si le fichier contient (triés par prix croissant) :

```
P0002,Souris,50,25.00
P0003,Clavier,30,45.75
P0001,Ordinateur,15,1200.50
```

    Et qu'on insère : Inserer_Trie(nomf, "P0005", "Webcam", 20, 89.99)

    Le fichier devient :

```
P0002,Souris,50,25.00
P0003,Clavier,30,45.75
P0005,Webcam,20,89.99
P0001,Ordinateur,15,1200.50
```


to be continued...