Rapport sur le Gestion d’un Stock de Produits

En Langage C

**EL JAOUHARI FAHD**

**SOUKRI OMAR**

**Année scolaire : 2024/2025**

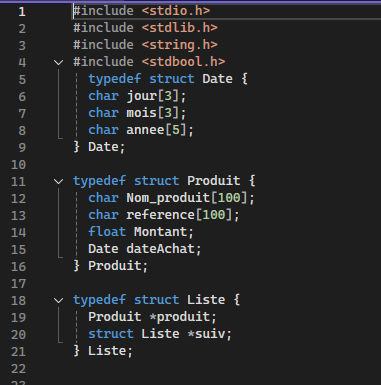
**2AP G9**

**INTRODUCTION**

Dans le domaine de la programmation, les listes chaînées représentent une structure de données fondamentale, permettant de gérer et de manipuler efficacement des ensembles d'éléments. Ce rapport se concentre sur l'implémentation et la gestion des listes chaînées en mettant en avant des méthodes clés et des structures utilisées pour leur création, leur manipulation et leur organisation.

À travers une approche détaillée, ce document explore différentes fonctions, telles que l'ajout, la suppression, le tri, ou encore la fusion des listes chaînées. Ces outils, souvent utilisés dans des applications informatiques complexes, constituent une base essentielle pour gérer des données dynamiques.

Ce rapport a pour objectif de fournir une compréhension claire et approfondie des listes chaînées, en illustrant chaque concept par des descriptions précises et une organisation méthodique des idées.



Ce code définit trois structures en C pour gérer des informations de produits et les organiser dans une liste chaînée :

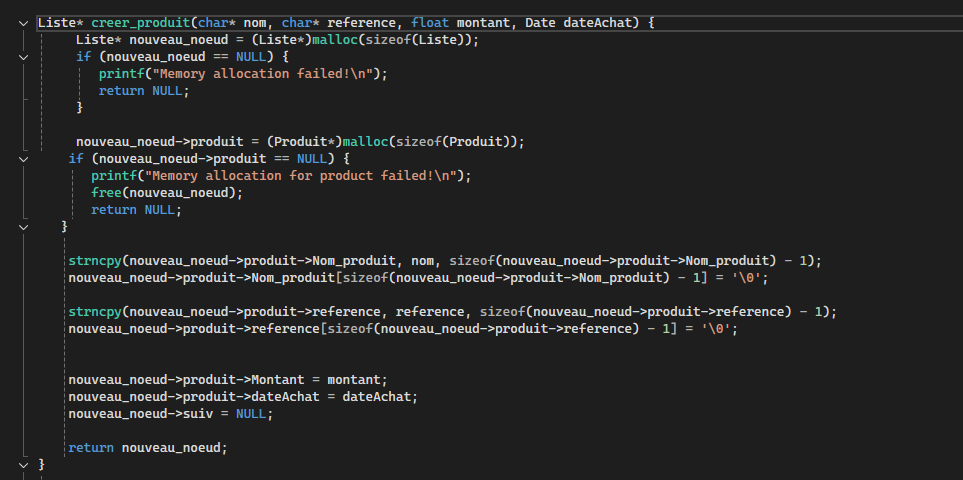
1. **Date :** Représente une date avec trois champs : jour, mois, annee, tous sous forme de chaînes de caractères.
2. **Produit :** Contient les informations sur un produit, incluant :

* Nom\_produit : le nom du produit.
* reference : la référence du produit.
* Montant : le prix du produit.
* dateAchat : la date d'achat du produit (type Date).

1. **Liste :** Représente un nœud dans une liste chaînée, avec :

* produit : un pointeur vers un Produit.
* suiv : un pointeur vers le nœud suivant dans la liste.

Cela permet de gérer une collection dynamique de produits dans une liste chaînée



Cette fonction creer\_produit crée un nouveau nœud pour une liste chaînée contenant un produit.

1. **Allocation mémoire :**

* Alloue de la mémoire pour un nœud de la liste (Liste\*) et pour un produit (Produit\*).
* Vérifie si l’allocation a échoué.

1. **Initialisation des champs :**

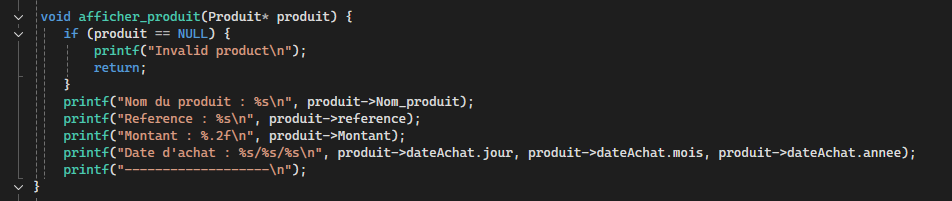
* Copie les valeurs (nom, référence, montant, date d'achat) dans le produit avec strncpy.
* Assure la terminaison des chaînes par un caractère nul (\0).

1. **Lien du nœud :**

* Initialise le pointeur suiv du nœud à NULL (fin de liste).

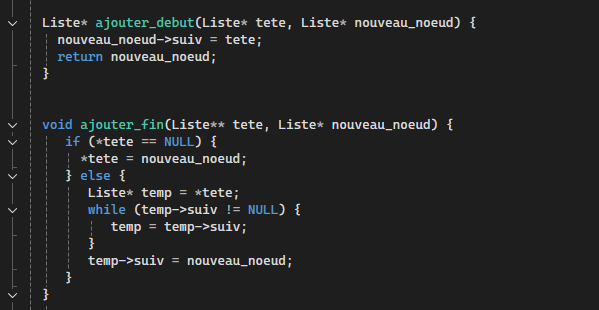
1. **Retourne le nœud :**

* Renvoie un pointeur vers le nœud nouvellement créé ou NULL en cas d’échec.



La fonction afficher\_produit affiche les détails d’un produit.

1. Vérifie si le pointeur du produit est valide (NULL) et affiche un message d'erreur si nécessaire.
2. Imprime les champs suivants du produit :
   * Nom.
   * Référence.
   * Montant (formaté avec 2 décimales).
   * Date d'achat (jour/mois/année).
3. Affiche une ligne de séparation.



ajouter\_debut :

* **But :** Insérer un nœud au début de la liste.
* **Fonctionnement :**

1. Le pointeur suiv du nouveau nœud est assigné à l'ancienne tête (tete).
2. Retourne le nouveau nœud comme nouvelle tête.

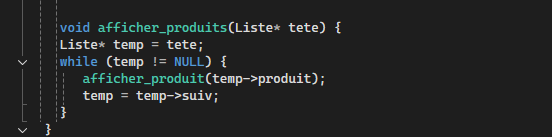
**Utilité :** Ajouter rapidement un élément en tête.

**ajouter\_fin :**

* **But :** Insérer un nœud à la fin de la liste.
* **Fonctionnement :**

1. Si la liste est vide (\*tete == NULL), le nouveau nœud devient la tête.
2. Sinon, parcourt la liste jusqu'au dernier nœud.
3. Relie le dernier nœud au nouveau via son pointeur suiv.

**Utilité :** Ajouter un élément à la fin de la liste.



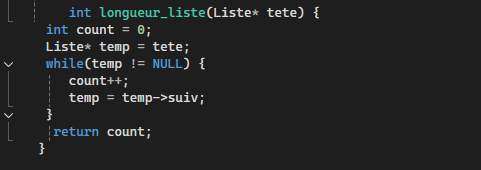
La fonction afficher\_produits affiche tous les produits d'une liste chaînée.

1. **Initialisation :** Utilise un pointeur temporaire temp pour parcourir la liste, initialisé à tete.
2. **Boucle :** Tant que temp n'est pas NULL :

* Appelle afficher\_produit pour afficher les détails du produit actuel.
* Passe au nœud suivant (temp = temp->suiv).

1. **Fin :** La boucle s'arrête après avoir parcouru tous les nœuds.

**Utilité :** Affiche proprement l'ensemble des produits d'une liste chaînée.



La fonction longueur\_liste calcule le nombre de nœuds dans une liste chaînée.

1. **Initialisation :**

* Initialise un compteur count à 0.
* Utilise un pointeur temporaire temp pour parcourir la liste, initialisé à tete.

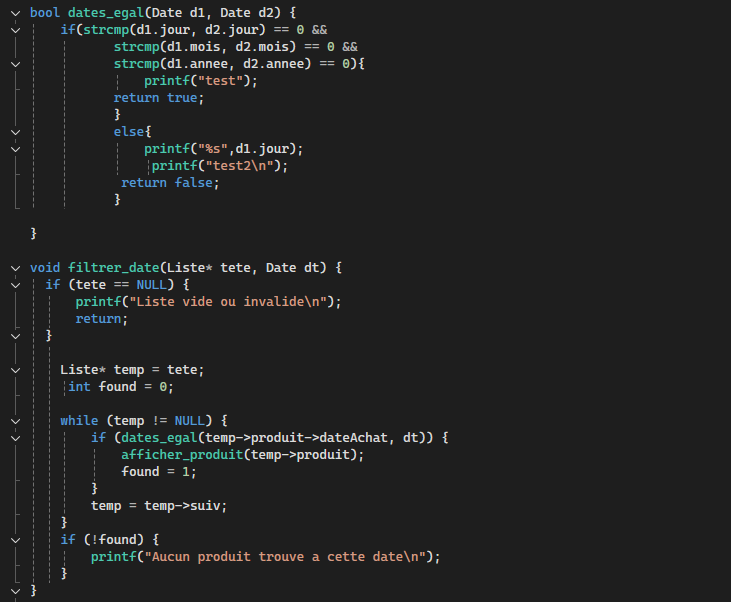
1. **Parcours :**

* Tant que temp n'est pas NULL :
* Incrémente le compteur count.
* Passe au nœud suivant (temp = temp->suiv).

1. **Retour :**

* Retourne le nombre total de nœuds comptés.

**Utilité :** Compter et retourner le nombre d'éléments dans une liste chaînée.



La fonction dates\_egal compare deux dates pour vérifier si elles sont identiques.

**Fonctionnement :**

* Compare les éléments jour, mois, et annee des deux dates avec strcmp.
* Si toutes les comparaisons réussissent, retourne true et affiche "test", sinon retourne false et affiche le jour de la première date avec "test2".

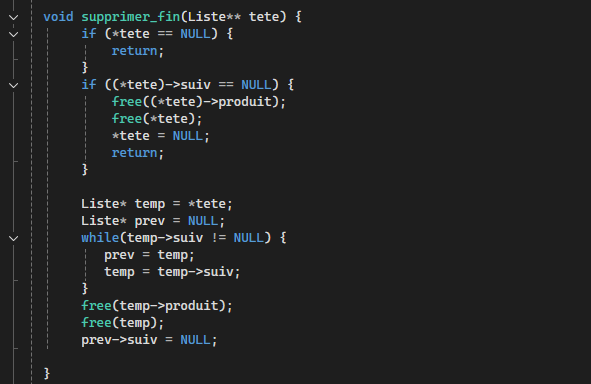
**Utilité** : Vérifier si deux dates sont égales.

La fonction filtrer\_date affiche les produits d'une liste chaînée ayant une date d'achat spécifique.

**Fonctionnement :**

* Parcourt la liste et compare chaque date d'achat avec la date donnée via dates\_egal.
* Affiche le produit si une correspondance est trouvée. Si aucun produit n'est trouvé, affiche un message.

**Utilité :** Filtrer et afficher les produits selon une date d'achat.



Voici un résumé des deux fonctions présentées :

supprimer\_debut(Liste\* tete) :

**Fonctionnement :**

* Si la liste est vide (tete == NULL), retourne NULL.
* Sinon, crée un pointeur newHead pointant sur le deuxième élément de la liste (tete->suiv).
* Libère la mémoire allouée pour le produit et le premier élément de la liste.
* Retourne le nouveau début de la liste (newHead).

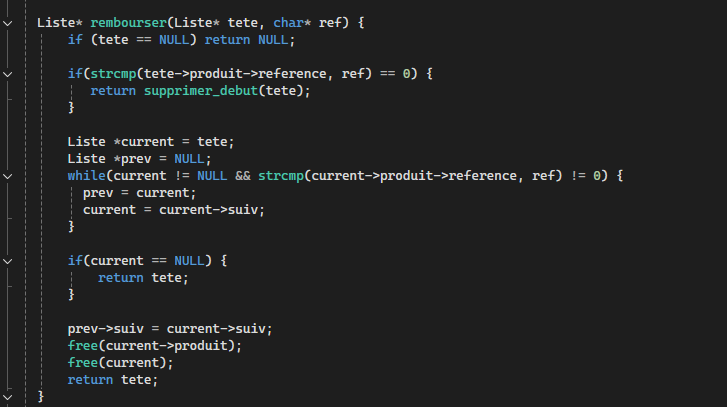
**Utilité :** Supprimer le premier élément de la liste chaînée.

supprimer\_fin(Liste\*\* tete) :

**Fonctionnement :**

* Si la liste est vide (\*tete == NULL), rien n'est fait.
* Si la liste contient un seul élément ((\*tete)->suiv == NULL), libère la mémoire du produit et de l'élément, puis met \*tete à NULL.
* Sinon, parcourt la liste jusqu'au dernier élément, libère la mémoire du produit et du dernier élément, puis met à jour le pointeur suiv du précédent élément pour qu'il pointe sur NULL.

**Utilité :** Supprimer le dernier élément de la liste chaînée.

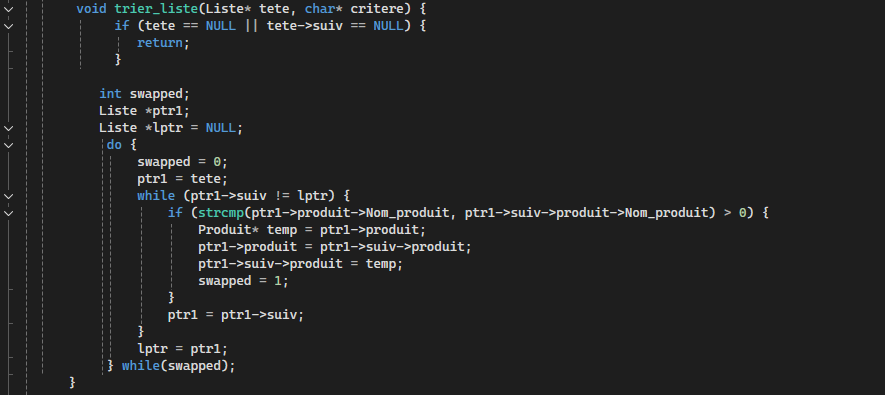


La fonction rembourser supprime un produit de la liste en fonction de sa référence.

**Fonctionnement :**

* Si la liste est vide, retourne NULL.
* Si le produit à la tête de la liste a la référence correspondante, appelle supprimer\_debut pour le supprimer.
* Sinon, parcourt la liste pour trouver le produit avec la référence donnée.
* Lorsque le produit est trouvé, il est supprimé de la liste, et la mémoire est libérée.
* Retourne la liste mise à jour.

**Utilité :** Supprimer un produit spécifique (par référence) de la liste chaînée.

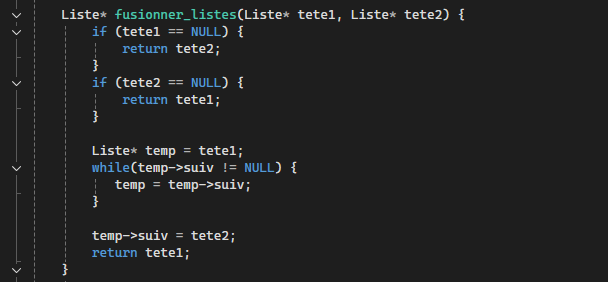


La fonction trier\_liste trie une liste chaînée de produits par ordre alphabétique du nom du produit.

**Fonctionnement :**

* Si la liste est vide ou contient un seul élément, rien n'est fait.
* Utilise l'algorithme de tri à bulles pour trier la liste :
* swapped est utilisé pour vérifier si un échange a eu lieu.
* Parcourt la liste, compare le nom des produits de deux éléments voisins (ptr1 et ptr1->suiv).
* Si les noms sont dans le mauvais ordre, les produits sont échangés.
* Le processus continue jusqu'à ce qu'aucun échange ne soit effectué, ce qui indique que la liste est triée.

**Utilité :** Trier les produits d'une liste chaînée par nom de produit de manière croissante.

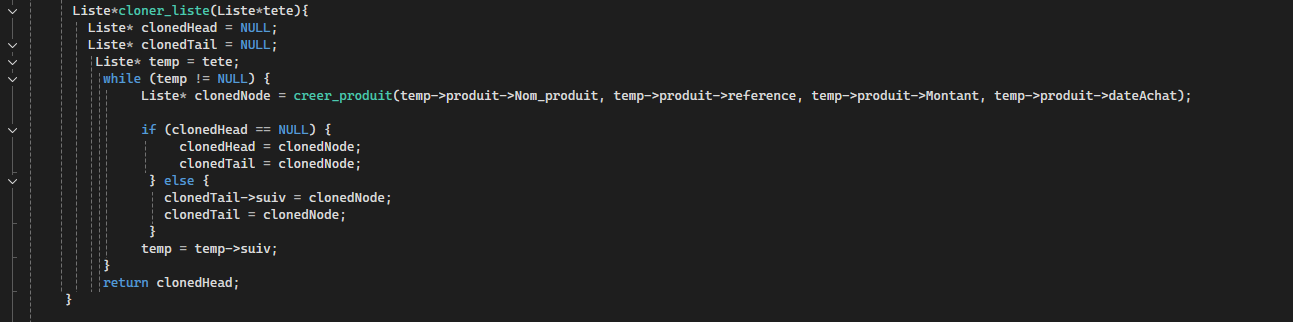


La fonction fusionner\_listes fusionne deux listes chaînées.

**Fonctionnement :**

* Si la première liste (tete1) est vide, elle retourne la deuxième liste (tete2).
* Si la deuxième liste (tete2) est vide, elle retourne la première liste (tete1).
* Si les deux listes sont non vides, elle parcourt la première liste jusqu'à son dernier élément, puis lie ce dernier élément au début de la deuxième liste.
* Retourne la tête de la première liste, maintenant fusionnée avec la deuxième.

**Utilité :** Fusionner deux listes chaînées en une seule, la deuxième liste étant ajoutée à la fin de la première.

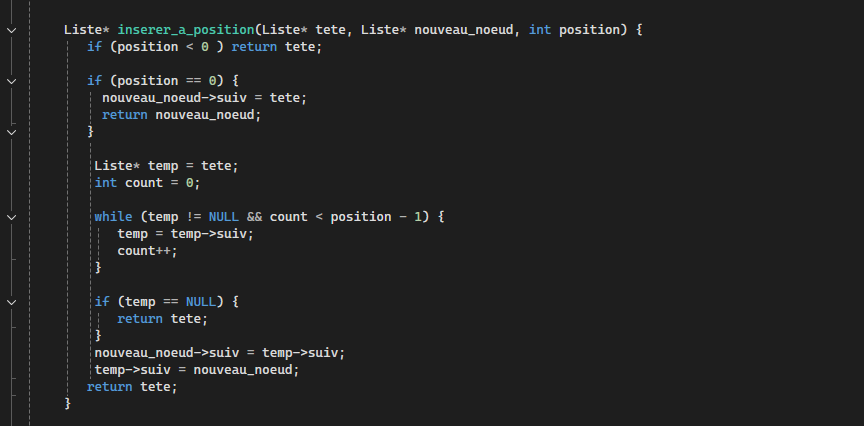


La fonction cloner\_liste crée une copie d'une liste chaînée.

**Fonctionnement :**

* Initialisation de deux pointeurs clonedHead et clonedTail à NULL pour suivre le début et la fin de la nouvelle liste.
* Parcourt chaque élément de la liste d'origine (tete), créant une copie de chaque produit avec creer\_produit.
* Si la liste clonée est vide, l'élément cloné devient à la fois la tête et la queue. Sinon, il est ajouté à la fin de la liste clonée.
* Retourne la tête de la liste clonée.

**Utilité :** Créer une copie complète d'une liste chaînée, y compris des nouveaux nœuds pour chaque produit.

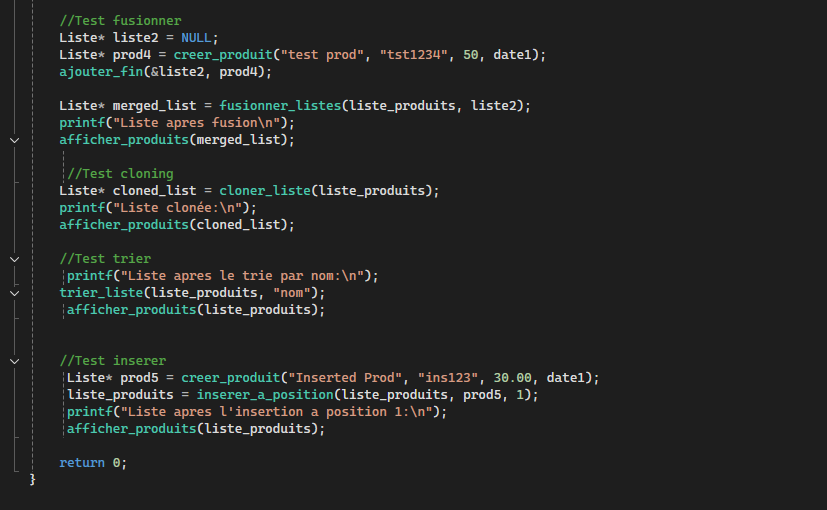
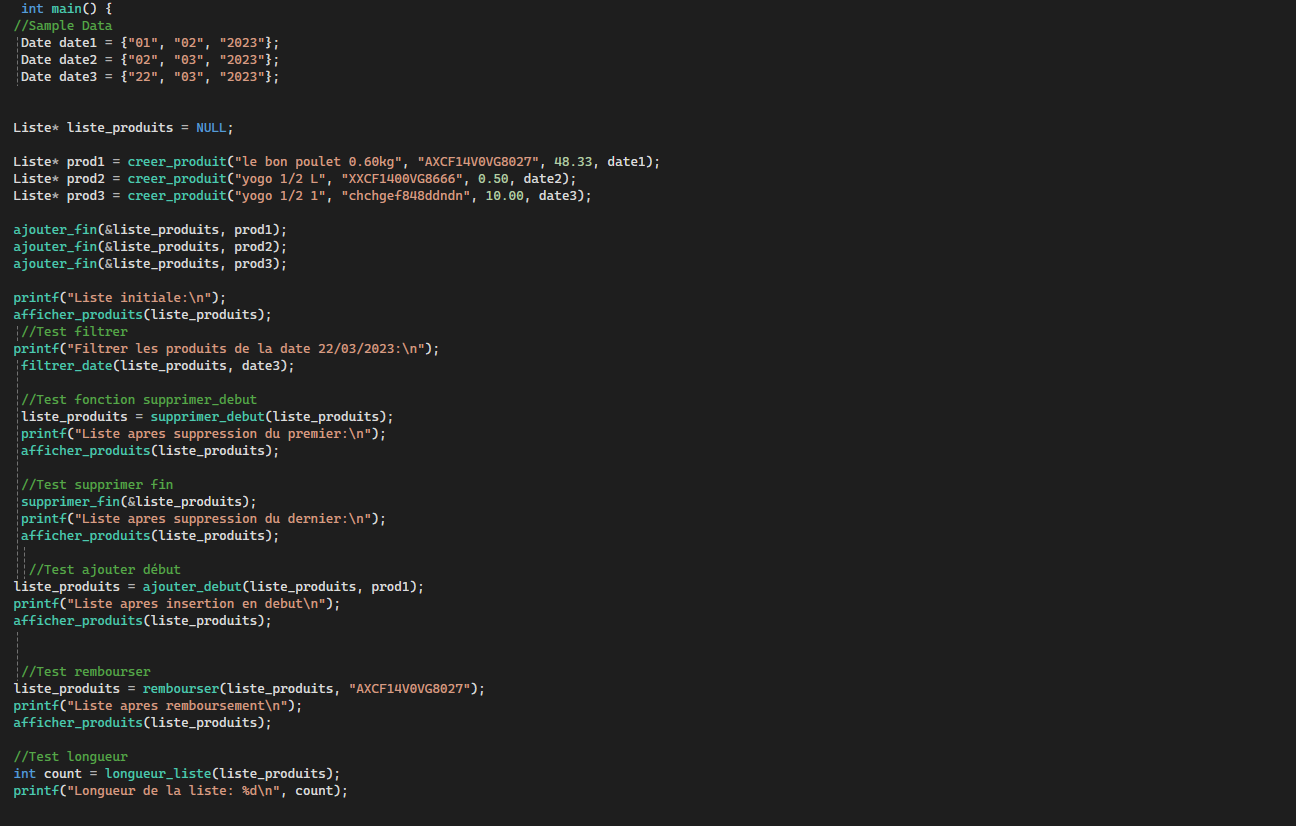


La fonction inserer\_a\_position insère un nouveau nœud à une position spécifique dans une liste chaînée.

**Fonctionnement :**

* Si la position est négative, retourne la liste inchangée.
* Si la position est 0, insère le nouveau nœud au début de la liste et le rend tête de la liste.
* Sinon, parcourt la liste jusqu'à la position souhaitée (position - 1).
* Si la position est valide, insère le nouveau nœud après le nœud correspondant à la position, en ajustant les pointeurs.
* Retourne la tête de la liste.

**Utilité :** Insérer un nouveau nœud à une position donnée dans une liste chaînée.



Le code dans main() teste plusieurs fonctions sur une liste chaînée de produits en utilisant des fonctions comme ajouter\_fin, filtrer\_date, supprimer\_debut, etc. Voici un résumé des étapes :

1. **Création de données d'exemple :**

* Trois dates et trois produits sont créés avec des informations comme le nom, la référence, le prix, et la date d'achat.

1. **Ajout de produits à la liste :**

* Les produits sont ajoutés à une liste chaînée vide à l'aide de ajouter\_fin.

1. **Filtrage des produits par date :**

* La fonction filtrer\_date est utilisée pour afficher les produits d'une date spécifique.

1. **Suppression du premier et du dernier produit :**

* supprimer\_debut supprime le premier produit, et supprimer\_fin supprime le dernier produit de la liste.

1. **Ajout d'un produit au début de la liste :**

* Un produit est ajouté au début de la liste avec ajouter\_debut.

1. **Remboursement d'un produit :**

* Le produit avec une référence spécifique est supprimé de la liste à l'aide de la fonction rembourser.

1. **Calcul de la longueur de la liste :**

* longueur\_liste renvoie le nombre d'éléments dans la liste.

1. **Fusion des listes :**

* Une nouvelle liste contenant un produit est fusionnée avec la liste existante à l'aide de fusionner\_listes.

1. **Clonage de la liste :**

* La fonction cloner\_liste crée une copie de la liste initiale.

1. **Tri de la liste par nom :**

* La liste est triée par nom de produit à l'aide de trier\_liste.

1. **Insertion d'un produit à une position spécifique :**

* Un produit est inséré à la position 1 de la liste avec inserer\_a\_position.

Ce programme teste donc l'ensemble des fonctionnalités de gestion de la liste chaînée de produits.

**EL JAOUHARI FAHD**

**SOUKRI OMAR**

**Année scolaire : 2024/2025**

**2AP G9**

**CONCLUSION**

Ce rapport a détaillé l'implémentation et l'importance des fonctions de gestion de liste en langage C. Grâce à ces fonctions, il est possible d'afficher, filtrer, trier, fusionner et manipuler des listes de produits efficacement. Les exemples concrets présentés démontrent leur utilité et leur application pratique. Ces outils sont essentiels pour optimiser la gestion des données et améliorer la performance des programmes développés en C. Ils offrent des solutions robustes et flexibles pour répondre aux divers besoins des utilisateurs.