

UNIVERSITE AFRICAINE DE TECHNOLOGIE ET MANAGEMENT

UATM – Département Génie Electrique



Domaine : Système Informatique et Logiciel (SIL)

Niveau : L1 – Génie Electrique

PROJET DE LANGAGE C

Thème du projet:

Logiciel de Gestion des Stocks d'une petite entreprise en mode console.

Réalisé par :

PEDROS MARCOS Samuel

Encadré par :

Mr KWAK-LIKAK Didier

Année universitaire : 2024 – 2025

PLAN

I- Introduction

II- Objectifs du projet

III- Structure et Explication brève du programme

IV- Difficultés rencontrées

V- Compétences acquises

VI- Conclusion

I-INTRODUCTION

Dans le cadre de notre apprentissage du langage C, nous avons réalisé un projet consistant à développer une application de gestion de stock pour une entreprise fictive. Ce programme fonctionne en mode console et permet la manipulation de produits à travers différentes opérations telles que l'ajout, la modification, la suppression, la recherche, la mise à jour de stock et l'enregistrement des données dans un fichier.

Ce projet nous permettra de mettre en pratique les bases du langage C, notamment les structures, les fichiers, les fonctions et les tableaux.

Projet Langage C - UATM

II- Objectifs du projet

Ce projet vise à atteindre les objectifs suivants :

- Concevoir une application simple de gestion de stock en langage C.
- Appliquer les notions fondamentales de la programmation procédurale.
- Gérer les données de manière persistante à l'aide de fichiers.
- Permettre une interaction utilisateur à travers un menu structuré.
- Mettre en place une logique de recherche, de mise à jour et d'affichage.

III- Structure et Explication brève du programme

Le programme est structuré de manière modulaire et comprend :

- Les directives d'inclusions qui importent des fonctionnalités prédéfinies que nous utiliserons dans notre programme et permettent d'utiliser des fonctions comme `printf()`, `scanf()`, `strcmp()`, `fopen()`, etc...

```
1  #include<stdio.h>
2  #include<windows.h> // Utilisé pour système("cls") et exit(EXIT_FAILURE)
3  #include<errno.h> // Pour errno
4  #include<string.h> // Pour strcmp, perror
```

- Les déclarations globales sont les données accessibles dans tout le programme, utilisées si plusieurs fonctions doivent accéder ou modifier les mêmes variables.

```
6  typedef struct{
7      char ref[20], designation[40];
8      short quantite;
9      float prix;
10 } produit;
11
12 produit p[200];
13 char ref[20];
14 short choix, quantite, nombre_produit, i,j,trouve = 0, ruptur = 0;
15 void ajouter_produit(), modifier_produit(), supprimer_produit(), rechercher_produit(), liste_produit(), enregistrer_produit(), mise_a_jour();
16 char reponse[4], answer[1];
```

- La fonction principale qui est le cœur du programme et contient le menu principal et les appels aux fonctions.

```
18
19 int main(){
20     do{
21         system("cls");//Permet d'effacer tout le contenu de l'écran
22         printf("***** MENU *****\n");
23         printf("1- Ajouter un produit.\n");
24         printf("2-Modifier un produit.\n");
25         printf("3- Supprimer un produit.\n");
26         printf("4- Rechercher un produit par référence.\n");
27         printf("5- Afficher la liste des produits.\n");
28         printf("6- Enregistrer les données dans un fichier texte ou binaire.\n");
29         printf("7- Mettre à jour les quantités après-vente ou approvisionnement.\n");
30         printf("8- Afficher les produits en rupture ou proche de rupture.\n");
31         printf("0- Quitter.\n");
32         printf("Choisissez une tâche: ");
33         scanf ("%hd", &choix);
34
35     switch(choix){
36         case 1: ajouter_produit();
37             break;
38         case 2: modifier_produit();
39             break;
40         case 3: supprimer_produit();
41             break;
42         case 4: rechercher_produit();
43             break;
44         case 5: liste_produit();
45             break;
46         case 6: enregistrer_produit();
47             break;
```

Projet Langage C - UATM

```
47     break;
48     case 7: mise_a_jour();
49         break;
50     case 8: rupture_produit();
51         break;
52     case 0: printf("Fin du programme");
53         break;
54     default:
55         printf("Choix invalide!\n");
56     }
57     printf("\nVoulez-vous continuer OUI/NON ? ");
58     scanf ("%3s", reponse); // Pas de & pour les tableurs
59 }
60 while(strcmp(reponse, "OUI") == 0);
61 return 0;
62 }
```

- Les fonctions qui servent à organiser le code en blocs réutilisables et chaque fonction effectue une tâche précise : ajouter, afficher, chercher, modifier, supprimer, enregistrer, mise à jour et montrer les produits en rupture de stock.

```
62 }
63
64 void ajouter_produit(){
77
78 void modifier_produit(){
102
103 void supprimer_produit(){
122
123 void rechercher_produit(){
142
143 void liste_produit(){
159
160 void enregistrer_produit(){
176
177 void mise_a_jour(){
204
205 void rupture_produit(){
219 }
```

A visualiser plus en détail dans le code source.

Voici une explication détaillée des principales fonctions :

- ✓ **ajouter_produit():** Cette fonction permet d'ajouter un nouveau produit au stock. L'utilisateur est invité à saisir la référence, la désignation, la quantité disponible et le prix unitaire du produit. Les informations sont ensuite enregistrées dans le tableau de structures, ce qui permet d'accroître le stock.
- ✓ **modifier_produit():** Elle permet de modifier les informations d'un produit existant, identifié à partir de sa référence. Si le produit est trouvé, l'utilisateur peut modifier sa désignation, sa quantité ou son prix.
- ✓ **supprimer_produit():** Cette fonction permet de supprimer un produit du stock. Après avoir recherché le produit par sa référence, le programme le retire du tableau en décalant tous les produits suivants d'une position vers la gauche. Cela libère la place occupée dans le tableau, et le nombre total de produits est décrémenté.
- ✓ **rechercher_produit():** Cette fonction permet à l'utilisateur de rechercher un produit à partir de sa référence. Si le produit est trouvé, ses informations sont affichées à l'écran. Sinon, un message d'erreur est affiché. Cette fonction repose sur une boucle de comparaison de chaînes de caractères.

Projet Langage C - UATM

- ✓ ***liste_produit()***: Elle affiche tous les produits enregistrés dans le tableau de stock. Pour chaque produit, les informations détaillées sont présentées (référence, désignation, prix, quantité).
- ✓ ***enregistrer_produit()***: Cette fonction permet de sauvegarder tous les produits enregistrés dans le tableau dans un fichier texte (Gestion_de_stock.txt). Les données sont écrites sous forme structurée dans le fichier, ce qui permet de conserver une trace du stock même après la fermeture du programme.
- ✓ ***mise_a_jour()***: Cette fonction permet d'ajuster la quantité d'un produit en stock. Une fois la référence saisie, l'utilisateur peut ajouter ou retirer une certaine quantité (dans le cas d'un approvisionnement ou d'une sortie de stock). La nouvelle quantité est ensuite affichée.
- ✓ ***rupture_produit()***: Elle identifie tous les produits dont la quantité est inférieure ou égale à un seuil critique (souvent ≤ 5). Ces produits sont considérés comme en rupture ou proches de la rupture de stock. Cette fonction est très utile pour alerter l'utilisateur sur les produits à réapprovisionner en priorité avant l'épuisement de son stock.

IV - Difficultés rencontrées 🧐

La réalisation de ce projet n'a pas été exempte de défis. Plusieurs difficultés ont été rencontrées tout au long du développement, qui nous ont permis de progresser sur le plan technique et qui étaient très difficiles à remarquer :

- ❖ La première difficulté a été la structuration correcte du programme, notamment la compréhension et l'organisation en quatre grandes parties : bibliothèques, variables globales, fonctions, et la fonction principale `main()`;
- ❖ Aussi de comprendre l'utilisation des nouvelles fonctions appris en dehors du cours;
- ❖ Une erreur fréquente a été l'utilisation incorrecte de variables globales comme `trouve` et `ruptur`, qui créaient des conflits de valeur entre les fonctions. Il a fallu apprendre à les remplacer par des variables locales pour rendre le code plus propre et sûr.
- ❖ Les erreurs de syntaxe comme l'appel à des fonctions non déclarées, les boucles mal formées ou encore les noms de fonctions mal orthographiés étaient difficiles à remarquer.
- ❖ L'affichage des messages comme "produit non trouvé" ou "aucune rupture", devait être déplacé en dehors des boucles pour ne pas s'afficher plusieurs fois;
- ❖ Le traitement correct des chaînes de caractères avec `scanf` ou `strcmp` a été une étape importante pour éviter les lectures incomplètes ou incorrectes et également hyper difficiles à agencer;
- ❖ C'était tout de même difficile de comprendre les messages d'erreur envoyées par le compilateur;
- ❖ Et enfin c'était difficile de savoir comment réorganiser le code pour mettre plus d'esthétique lors de l'affichage dans la console.

V- Compétences acquises 🎓

Malgré les obstacles rencontrés, ce projet a été **extrêmement formateur**. Chaque difficulté surmontée a contribué à enrichir nos connaissances en programmation C et à développer des réflexes de développeur rigoureux. Voici les principales compétences acquises :

- ✚ Structurer correctement un programme en C (bibliothèques, variables, fonctions, main)
- ✚ Créer et utiliser mes propres fonctions en dehors du cours
- ✚ Remplacer les variables globales par des variables locales pour éviter les conflits
- ✚ Lire et interpréter les messages d'erreur du compilateur
- ✚ Corriger des erreurs de syntaxe et de logique
- ✚ Maîtriser la gestion des chaînes de caractères avec `scanf` et `strcmp`
- ✚ Organiser proprement l'affichage dans la console
- ✚ Lire et écrire des données dans un fichier texte
- ✚ Développer une autonomie et une rigueur dans la correction des bugs

VI- CONCLUSION

La réalisation de ce projet de gestion de stock en langage C a été une expérience à la fois enrichissante et formatrice. Malgré les nombreuses difficultés rencontrées, chaque étape du développement m'a permis de renforcer mes compétences techniques, de mieux structurer mon raisonnement, et surtout de comprendre l'importance de la rigueur en programmation. Ce travail m'a donné plus d'assurance dans l'écriture de programmes complets et fonctionnels.

Je tiens à remercier sincèrement **notre professeur** pour ses enseignements, sa patience et ses conseils précieux, qui m'ont permis de mener ce projet à bien. Ce projet marque un tournant important dans ma progression et restera une base solide pour les prochains défis en programmation.