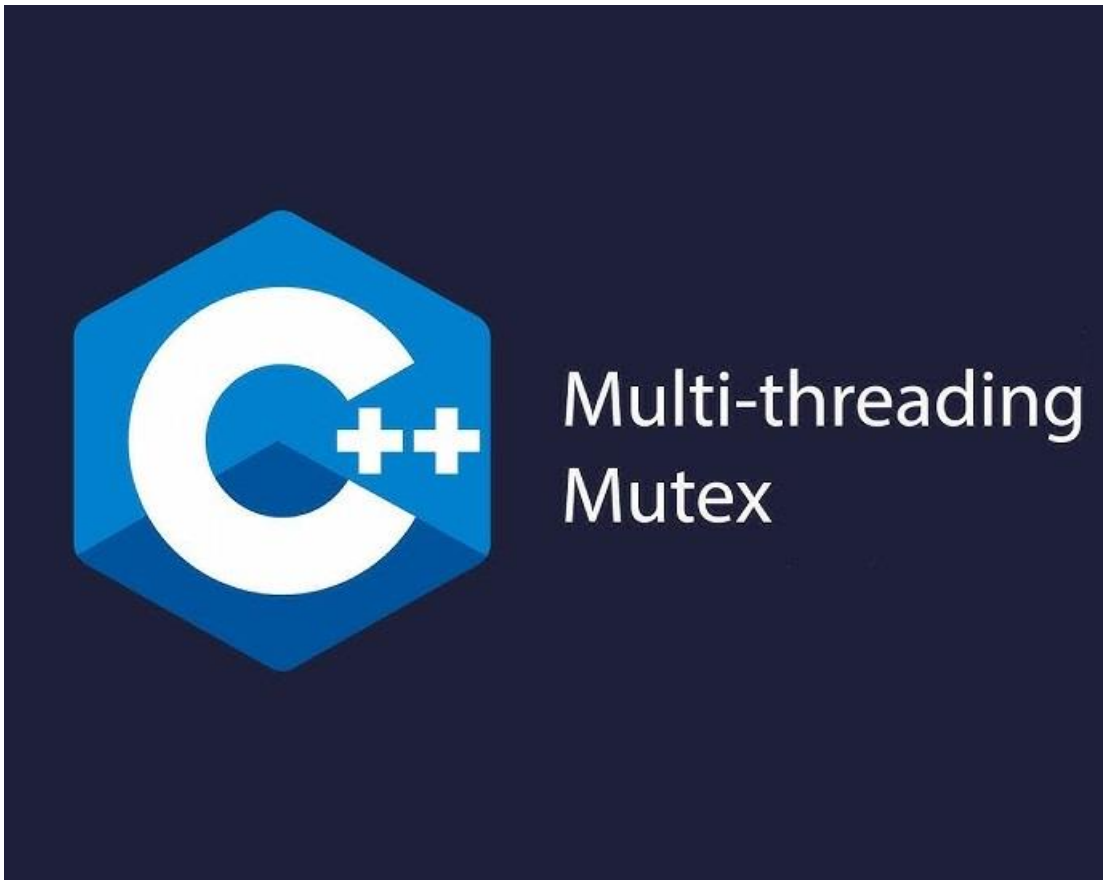
Labo 3 - Gestion d’accès concurrents



Durée du travail :	**Du 18.10.2023 au 07.11.2023**
Auteurs :	***Kilian Demont*** **Calvin Graf**
Enseignant :	**Florian Vaussard**
Cours :	**PCO**
Lieu de travail :	**HEIG-VD, Yverdon-les-Bains**

Table des matières

1. Introduction

Ce rapport présente un projet de simulation des chaînes de production de robots, depuis la matière première jusqu'au produit. Le laboratoire porte sur la gestion des transactions entre les différents acteurs et donc la bonne gestion des fonds et des inventaires de chaque intervenant dans le marché.

L'objectif de cette simulation est de comprendre comment ces acteurs interagissent pour produire et distribuer des produits, tout en gérant leurs ressources financières et matérielles. Notre travail se concentre autour de cet aspect et vise à nous faire comprendre les notions de threads, sections critiques et de verrous. Les autres aspects du projet (à l'exception de la terminaison des threads) comme l'UI sont déjà implémentés.

2. Développement

2.1 Architecture du projet

Le projet est structuré en plusieurs classes, chacune représentant un type d'acteur. Les classes principales incluent :

Extractor : Modélise les entités qui extraient des matières premières telles que le cuivre, le sable et le pétrole.

Factory : Représente les usines qui transforment les matières premières en produits finis tels que les puces électroniques, le plastique et les robots.

Wholesale : Modélise les grossistes qui achètent produits des usines ainsi que des matières premières qu'ils revendent.

Seller : Modélise les vendeurs et fournis une structure commune pour la gestion des stocks, des fonds et des transactions. Extractor, Factory, Wholesale dérivent de cette classe et hérite de ses propriétés.

Utils : Gère l'initialisation, la simulation et la fin du programme.

MainWindow : Fournit l'interface utilisateur pour afficher les informations de simulation en temps réel.

2.2 Choix

Lors de l'achat de matériel d'une usine, nous avons pris la décision d'acheter les produits un par un. Puisque chaque usine a besoin d'une ressource unique, il est inutile de se précipiter sur l'achat de ressource. Ainsi, chaque usine va récupérer les ressources dont il a besoin une par une pour créer son produit et le revendre. Cela évite par exemple d'acheter 5 unités d'une ressource, et de ne plus avoir assez pour la deuxième ressource nécessaire et de se retrouver bloqué.

2.3 Utilisation de la concurrence et des mutex

Un aspect essentiel de ce projet était de gérer la concurrence, car de nombreux acteurs tentent d'accéder aux mêmes ressources, telles que les matières premières (stocks) et les fonds (money). Pour ce faire, des verrous (mutex) ont été utilisés pour garantir l'accès sécurisé à ces ressources partagées. Extractor, Factory et Wholesale ont leurs propres mutex hérités de la classe Seller pour protéger ses données internes contre les accès concurrents. Cette approche assure que les données sont manipulées de manière cohérente et évite les conditions de concurrence.

Pour l'utilisation des mutex dans la fonction "orderRessources()" de "factory.cpp". La fonction étant relativement compliquée et une bonne partie du code ayant besoin de mutex. Nous avons décidé de protéger l'intégralité de la fonction afin de ne pas complexifier encore plus le code avec de multiples mutex. Le coût de cette opération est moindre et simplifie grandement la lecture de la fonction.

2.4 Tests

Nous avons effectué une batterie de tests afin de vérifier le bon fonctionnement de notre programme. Afin d'être sûr que chaque opération a bien été réalisée, nous avons évidemment vérifié si les stocks et l'argent affichée sur l'interface graphique étaient correctes et si le message de fin nous confirmait que nous avions la même quantité d'argent qu'au lancement du programme.

Mais nous avons également utilisé les « interface->consoleAppendText » en affichant des messages et des variables pour confirmer que les valeurs étaient correctes.

Voici toutes les vérifications que nous avons réalisées :

Extracteurs

- Ils minent les matières premières : OK
- Ils paient les employés qui minent : OK
- Ils vendent leurs ressources aux grossistes et récupèrent l'argent : OK

Grossistes

- Ils achètent entre 1 et 5 ressources aux extracteurs de manière aléatoire : OK
- Ils achètent entre 1 et 5 ressources aux usines de manière aléatoire : OK
- Ils vendent leurs ressources aux usines lorsqu'une demande arrive : OK

Usines

- Elles achètent les ressources qu'elles ont besoin et dont le stock est le plus bas 1 par 1 aux grossistes : OK
- Elles fabriquent des objets dès que les ressources sont disponibles : OK
- Elles paient les employés pour la création d'objet : OK
- Elles vendent leurs objets fabriqués aux grossistes : OK

Programme

- Le nombre de pièces de départ initié à 2000 ne change pas entre le début et la fin de l'exécution : OK
- Lorsqu'un acteur n'a plus de pièces, il n'achète plus et ne fabrique plus : OK
- Lors de la fermeture du programme, le message de fin s'affiche et le programme se ferme correctement : KO

2.5 Bug à régler

Lors de la phase de test nous avons décelé un bug que nous n'avons malheureusement pas su résoudre. Lorsque nous souhaitons fermer notre programme, il arrive que celui-ci plante et ne se ferme pas correctement.

Si la fermeture a lieu au milieu de l'exécution, il n'y a aucun problème. Cependant, si nous le fermons lorsque nous arrivons à la fin du programme (c'est-à-dire quand les acteurs sont bloqués par manque de moyen), alors une erreur se produit et l'application s'arrête sans afficher le message de fin.

3. Conclusion

La simulation de chaînes de production nous a permis d'utiliser et d'acquérir la compréhension de concepts liés à la gestion d'accès concurrents. L'utilisation de la concurrence et des mutex a assuré un accès sécurisé aux ressources partagées telles que les stocks et les fonds des différents acteurs. Les tests ont mis en évidence le bon fonctionnement de la gestion d'accès concurrents mais également celui d'un bug qui devra être résolu pour garantir la stabilité du programme à la fermeture.