Workshop - C++:

Application de Gestion de Matériel

Préparé par CPU ISIMM

Date :
Novembre 2024

1 Introduction

Objectif: Le but de ce workshop est de développer une application de gestion de matériel pour notre Responsable Technique, Fehd Ferhi.

Contexte: Le club CPU ISIMM, spécialisé en robotique, compte un grand nombre de membres, actifs et seniors. La gestion des ressources robotiques est donc complexe pour notre responsable technique, Fehd, qui doit gérer efficacement les équipements prêtés aux membres.

Mission: Développer une application en C++ qui assiste Fehd dans la gestion des membres du club et du matériel, en fournissant des fonctionnalités spécifiques pour faciliter l'organisation et la distribution des composants robotiques.

2 Développement de l'application

L'application sera développée étape par étape.

2.1 Étape 1: Gestion des Membres

1. **Définir la structure 'Membre':** Chaque membre de CPU ISIMM possède un: - `nom` : Nom du membre, - `prénom` : Prénom du membre, - `id` : Identifiant unique, - `role` : Statut (actif ou senior).

2. **Créer une fonction `createMember`:** Cette fonction permet de créer un nouveau membre. Elle prend les informations de l'utilisateur et retourne la structure 'Membre' créée.

3. **Ajouter plusieurs membres avec `addManyMembers`:** Cette fonction prend en paramètres un tableau dynamique de membres et un entier `n` représentant le nombre de membres à ajouter. Elle utilise `createMember` pour ajouter `n` membres au tableau.

4. **Supprimer le dernier membre avec `removeLastMember`:** Cette fonction permet de supprimer le dernier membre ajouté dans la liste.

5. **Supprimer un membre par `id` avec `removeMemberById`:** Cette fonction prend en paramètre l'identifiant `id` d'un membre ,le tableau des membres et supprime ce membre du tableau s'il existe.

6. **Afficher les informations d'un membre avec `displayMemberById`:** Cette fonction prend en paramètre l'`id` d'un membre, le tableau des membres et affiche ses informations.

2.2 Étape 2: Recherche et Affichage des Membres

Pour faciliter la gestion des membres, Fehd souhaite les fonctionnalités suivantes:

1. **Rechercher un membre par `id` avec `searchById`:** Cette fonction prend en paramètre le tableau de membres et un `id`. Elle retourne un booléen indiquant si le membre existe.

2. **Afficher les membres par prénom avec `displayMembersByFirstName`:** Cette fonction prend en paramètre le prénom recherché et affiche la liste des

membres avec ce prénom. Si aucun membre ne correspond, un message Pas de membre avec ce prénom est affiché.

2.3 Étape 3: Gestion des Composants

Pour gérer efficacement les composants électroniques et les allocations aux membres, nous utiliserons: - Une `map` où chaque clé est l'identifiant unique d'un composant, et la valeur est une paire `(quantité totale, quantité allouée)`. - Un `set` pour stocker les membres qui ont alloué des composants. Chaque élément du `set` est une paire où le premier élément est l'`id` du membre et le second est le nombre de composants alloués.

2.3.1 Fonctions de Gestion des Composants

1. **Ajouter un nouveau composant avec `addComposantFirstTime`:** Cette fonction prend de l'utilisateur l'`id` du composant et le `totalQuantity`. Elle initialise `allocatedQuantity` à 0 et ajoute ce composant dans la `map`.

2. **Ajouter des quantités supplémentaires avec `addComposant`:** Cette fonction prend un `id` de composant et une quantité supplémentaire. Elle incrémente la quantité totale du composant dans la `map`.

3. **Vérifier la disponibilité d'un composant avec `composantIsAvailable`:** Cette fonction vérifie si le composant existait (par ID passé en paramètres). Si c'est le cas , on veréfiera si la quantité disponible (quantité totale - quantité allouée) est suffisante pour l'allocation demandée. Elle retourne un booléen.

2.3.2 Fonctions d'Allocation de Composants aux Membres

1. **Allouer un composant avec `allocateComposant`:** Cette fonction permet à un membre d'allouer un composant (1 composant maximum par membere), en vérifiant d'abord sa disponibilité. Si les conditions sont remplies(le membre n'a jamais alloué un composant et le composant est disponible), elle met à jour la quantité allouée et ajoute le membre dans le `set` des membres avec allouement.

2. **Vérifier si un membre a déjà alloué avec `isMemberAllocated`:** Cette fonction permet de vérifier si un membre a déjà alloué un composant.

3. **Afficher les membres avec composants alloués avec `displayAllocatedMembers`:** Cette fonction affiche tous les membres ayant alloué un composant, en affichant leurs informations. Elle utilise les informations du `set` pour générer la liste.

3 Script Principal

Le script principal (`main`) de l'application organise les étapes suivantes: - Initialiser les structures de membres et de composants. - Appeler les fonctions définies pour ajouter, supprimer, afficher, et allouer les composants aux membres. - Gérer l'interaction avec l'utilisateur pour entrer des données et naviguer dans les différentes options.