INF2610

TP #1 : Le langage C

Polytechnique Montréal

Hiver 2024
Date de remise: Voir le site Moodle du cours
Pondération: 2,5%

Description	Note	Total
Partie 1: développement des pièces		4
Partie 2: développement de la fonctionnalité de création des avions		10
Partie 3: Fonctions		6
		20

Prenez le temps de lire attentivement l'énoncé.

Objectifs

Les travaux pratiques ainsi que la majeure partie du cours INF2610 *Noyau d'un système d'exploitation* vous amèneront à lire et créer des programmes en langage C. La réussite des travaux pratiques nécessite la maîtrise de certaines notions fondamentales liées à ce langage. Dans le cadre de ce premier travail pratique, nous allons revenir sur les notions suivantes:

- La syntaxe du langage C
- Les `struct`
- Les pointeurs
- L'allocation dynamique

Cette séance introductive vise à vous permettre, lors des séances subséquentes, de vous concentrer sur les enjeux spécifiques à la thématique du cours, plutôt que sur ceux liés au langage.

Contexte

Vous êtes en stage au sein de l'entreprise AirPoly. On vous demande de compléter le fichier **Lab.c**. L'équipe vous invite à tester votre code par vous-même à l'aide des commandes suivantes :

1. `make`
2. `./lab`

Votre mentore vous invite à prendre connaissance du design détaillé fourni dans les sections suivantes. Vous devez vous baser sur ces informations pour programmer les fonctionnalités manquantes dans **Lab.c**.

Travail demandé

Partie 1: développement des pièces — 4 points

Vous devez créer les trois `struct` suivantes:

- `Plane`
- `Wheel`
- `Wing`

Les attributs et leurs types pour chaque `struct` sont définis comme suit. *Si vous n'avez pas accès à un type primitif, vous devez l'inclure dans le fichier.*

`struct Plane`:

`id`, un tableau de caractères

`planeType`, une chaîne de caractères de dimension 10

`isAvailable`, un booléen

`wheels`, un tableau de type `struct Wheel`

`wings`, un tableau de type `struct Wing`

`struct Wheel`:

`id`, un entier

`isRearWheel`, un booléen

`struct Wing`:

`id`, un tableau d'entiers

Partie 2: développement de la fonctionnalité de création des avions — 10 points

Vous devez ensuite implémenter les fonctions suivantes pour créer un `Avion`.

Créer une roue — 2 points

La fonction `createWheels` devrait générer un tableau de nouvelles roues à l'aide du paramètre initial `id`. Cette fonction devrait retourner un tableau de sept roues. La première roue aura comme identifiant la valeur de `id`, et chaque roue subséquente aura comme identifiant l'incrément de celui de la roue précédente. Par exemple, si la fonction reçoit `id = 101`, la première roue aura `id = 101`, la seconde aura `id = 102`, la troisième aura `id = 103`, et ainsi de suite. L'attribut `isRearWheel` devrait valoir `false` pour les trois premières roues et `true` pour les autres.

Créer une aile — 4 points

La création d'une aile nécessitera deux fonctions distinctes. D'abord, la fonction `populateWingAttributes` va prendre comme paramètres une `struct Wing`, ainsi qu'un entier `id` qui servira à construire l'identifiant de l'aile. La fonction devrait allouer **dynamiquement** un tableau de neuf entiers dont les valeurs seront ensuite transférées de manière appropriée dans l'attribut `id` qui appartient à l'aile. Chaque élément du tableau alloué dynamiquement correspondra à un chiffre de l'identifiant. Lorsque l'identifiant comporte moins de neuf chiffres, les éléments du tableau qui ne sont pas utilisés doivent être initialisées à 0. Pour la logique de placement dans le tableau, référez-vous aux exemples à la prochaine page.

Exemple : id = 123456

0	0	0	0	1	2	3	4	5	6

Exemple : id = 54321

0	0	0	0	0	5	4	3	2	1

Exemple : id = 123

0	0	0	0	0	0	0	1	2	3

La fonction `populateWingAttributes` *ne comporte aucune valeur de retour*.

La fonction `createWings`, va prendre en paramètre une variable `id` de type `long`, procèdera à l'allocation dynamique de deux ailes qui seront ensuite retournées à l'appelant. Le mécanisme d'assignation incrémentale des identifiants à chaque aile à partir de la valeur passée en paramètre **est identique à celui des roues**. La fonction *doit* :

- Utiliser `populateAttributes`; et
- Retourner un tableau de `struct Wing` .

Créer un avion — 4 points

La fonction `createPlanes` reçoit en paramètres un pointeur vers un tableau de `struct Plane`, un pointeur vers un tableau de **caractères** contenant l'identifiant de l'avion, un entier contenant le nombre d'avions à créer. L'attribut `planeType` doit être initialisé comme tableau vide. L'attribut `isAvailable` doit être initialisé à `true`. La fonction doit utiliser les fonctions `createWings` et `createWheels` pour populer les attributs `wheels` et `wings`. Elle *ne comporte pas de valeur de retour*.

Partie 3: développement des fonctionnalités supérieures — 6 points

Changer la disponibilité d'un avion — 1 point

La fonction `setAvailability` reçoit en paramètre un pointeur vers un avion, ainsi qu'un booléen. Elle modifie l'attribut `isAvailable` de l'avion avec le booléen. *La fonction ne retourne rien.*

Retourner les avions disponibles — 1 point

La fonction `getAvailablePlanes` reçoit un pointeur vers un tableau d'avions et le nombre d'avions dans la liste pour retourner les identifiants des avions qui sont disponibles.

Classifier les avions — 2 points

La fonction `setPlaneType` reçoit en paramètre un pointeur vers un avion. *Elle ne comporte pas de valeur de retour.* Elle affichera la classification de l'avion en se basant sur le modulo de la valeur de l'identifiant de la *première* aile de l'avion :

`identifier` $\in [0,2] \implies$ "Small"

`identifier` $\in [3,6] \implies$ "Medium"

`identifier` $\in [7,8] \implies$ "Large"

Bien entendu, la fonction doit reconstruire l'entier contenu dans le tableau avant de calculer son modulo.

Retourner les avions d'un type donné — 2 points

La fonction `getPlanesByType` prend en paramètre un pointeur vers un tableau d'avions, un tableau de caractères qui représente le type que l'on recherche et le nombre d'avions dans la liste. On vous demande d'utiliser la fonction `strcmp` pour évaluer si le type de chacun des avions est identifique à celui passé en paramètre. La fonction retournera un tableau contenant les avions qui correspondent au type recherché.

Remise

Instructions

À la fin du TP, lancez la commande `make handin` dans le répertoire principal du TP afin de créer l'archive `handin.tar.gz` que vous devez par la suite renommer avec vos matricules. Vous pourrez ensuite remettre votre le fichier dans la boîte de remise prévue à cet effet sur le site Moodle du cours.

⚠La lisibilité de votre code est cruciale, et les lacunes à cet égard seront pénalisées.⚠

Dates limites et politique de retards

Le travail doit être remis avant le début de votre prochaine période de laboratoire. Vous aurez 2 semaines pour le compléter. Les retards ou oublis sont sanctionnés de 25% pour tout retard de 24h ou moins, 50% pour tout retard entre 24h et 48h, et 100% pour tout retard de plus de 48h.

Correction

Vous êtes limités **à la version C11**. Tout code qui ne respecte pas cette contrainte sera pénalisé. Assurez-vous de suivre les instructions clairement et respecter les critères de paramètres et retours de fonctions.