**Langage de programmation avancé**

**HAUTE ÉCOLE DE NAMUR-LIÈGE-LUXEMBOURG**

**DA-IA – bloc 1**

Exercices 1 – Structures

Objectifs

* Continuer à utiliser les éléments déjà vus
* Continuer les bonnes pratiques acquises lors des exercices précédents
* Apprendre à écrire et à utiliser des structures

# Inventaire

Le but de cet exercice est de gérer l’inventaire d’un personnage qui peut collecter et déposer des objets.

Commencez par créer un nouveau projet (et solution) en lui donnant comme nom Inventaire, par exemple. N’oubliez pas de faire en sorte que le fichier principale, celui qui contient la fonction main, ait l’extension .c.

Lisez ensuite attentivement l’énoncé et assurez-vous d’avoir compris la structure des données et les fonctions à définir.

## Contexte

L’inventaire est représenté par 9 cases (nombre qui devrait pouvoir être changé facilement). Ces cases peuvent contenir chacune soit un objet soit un groupe de plusieurs objets de même type. On suppose que les cases occupées sont toujours les premières : si on vide une case qui contenait un objet, les objets des cases suivantes glissent automatiquement vers la gauche.



On utilise un référentiel sous la forme d’un tableau décrivant les types d’objets existants.

Pour chaque type d’objet, on retient

* une référence numérique (numéro d’identification de l’objet),
* un nom et
* un poids en grammes.

Un inventaire est décrit par

* le nombre de cases remplies et
* un tableau reprenant, pour chaque case, la référence de l’objet et le nombre d’objets contenus dans la case.

## Étape 1 : définir les structures

1. Définissez une structure d’étiquette objet pour représenter les types d’objets ainsi qu’un synonyme.
2. Définissez une structure d’étiquette pileObjets pour représenter le contenu d’une case de l’inventaire ainsi qu’un synonyme.
3. Définissez une structure d’étiquette inventaire pour représenter un inventaire, ainsi qu’un synonyme.

## Étape 2 : définir le référentiel des objets

Dans la fonction principale, déclarez et initialisez le tableau objets contenant les descriptions des types d’objets.

Ce tableau est passé en argument à toutes les fonctions qui en auront besoin.

Vous pouvez utiliser le code suivant pour initialiser le tableau en question une fois celui-ci déclaré.

{

{ 117, "arc", 1000 },

{ 135, "epee courte", 1000 },

{ 57, "fleche", 25 },

{ 10, "torche", 500 },

{ 215, "pelle", 4000 },

{ 202, "grappin", 2000 },

{ 194, "seau", 1000 }

}

## Étape 3 : définir un début d’inventaire

Dans la fonction principale, pour pouvoir tester les fonctions suivantes, copiez le code ci-dessous afin de définir une variable représentant l’inventaire en reprenant les informations de l’image ci-dessus (comportant donc 1 arc, 7 torches, 1 épée courte et 32 flèches).

Inventaire inventaire = {

4,

{

{117,1},

{10,7},

{135,1},

{57,32}

}

};

## Étape 4 : rechercher un objet dans le référentiel

Définissez une fonction qui, recevant en autre une référence numérique d’objet, renvoie soit l’indice dans objets correspondant à l’objet en question (s’il existe), soit -1 (si l’objet n’existe pas). N’oubliez pas te tester votre code avant d’aller plus loin !

Par exemple, en considérant le référentiel décrit à l’étape 2,

* si on recherche l’objet dont la référence est 57, la fonction renvoie l’indice 2 ;
* si on recherche l’objet dont la référence est 42, la fonction renvoie -1.

## Étape 5 : afficher l’inventaire

Définissez une fonction qui affiche un inventaire donné au format suivant :  
1 × arc  
7 × torche  
1 × epee courte  
32 × fleche  
Total : 6300 g

À nouveau, testez votre code avant de continuer.

## Étape 6 : obtenir une pile d’objets valide

Définissez une fonction dont l’objectif est d’obtenir de l’utilisateur une référence d’objet valable et un nombre d’objets valable ( 1).

Les informations, une fois validées, seront renvoyées sous la forme d’une structure pileObjets.

Pour valider la référence, pensez à utiliser la fonction de l’étape 4.

## Étape 7 : ajouter une pile d’objets à l’inventaire

Définissez une fonction qui reçoit un inventaire et une pileObjets correspondant à un ou plusieurs objets trouvés et se charge d’ajouter l’objet (ou les objets) trouvé(s) à l’inventaire et renvoie ce dernier, mis à jour, si c’est possible.

Plusieurs consignes sont à prendre en compte :

* Si une des cases de l’inventaire contient déjà des objets du même type que ceux qui sont à ajouter, ces derniers sont ajoutés à cette case.
* Sinon, s’il y a au moins une case libre, les objets se retrouvent dans la 1re case libre.
* Dans ces deux premiers cas, on affiche le message « Objets ajoutés ».
* Par contre, s’il ne reste plus de case libre, les objets ne sont tout simplement pas ajoutés à l’inventaire et un message « Inventaire plein » est affiché à l’écran.

## Étape 8 : créer un test complet

Lorsque vous rédigez du code, il est important de tester les différents cas qui peuvent se présenter. Par exemple, pour ajouter une pile, il faut tester les 3 cas suivants :

* un cas où ce n'est pas possible car les 9 cases de l’inventaire sont occupées par d’autres objets,
* un cas où la pile d’objets peut se combiner avec une pile existant déjà dans l’inventaire,
* un cas où ce type d’objets n’est pas encore présente dans l’inventaire, il faut donc mettre la pile d’objets dans une nouvelle case.

Pour tester le cas de l’inventaire plein, le plus simple est de réduire à 4 le nombre maximal de cases dans l’inventaire. En reprenant l’inventaire de l’étape 3, tentez d’ajouter un nouvel objet. Le message « Inventaire plein » devrait apparaître.

À vous de jouer pour les deux autres cas (n’oubliez pas de remettre le nombre maximal de case dans l’inventaire à 9).

## Étape 9 : déposer un objet

Définissez une fonction qui reçoit un inventaire et une pileObjets correspondant à un ou plusieurs objets à déposer et se charge d’enlever l’objet (ou les objets) de l’inventaire puis de renvoyer ce dernier, mis à jour, si c’est possible.

Plusieurs consignes sont à prendre en compte :

* On suppose qu'il n'y a pas deux cellules d'inventaire occupées par le même type d'objets.
* Si l’inventaire contient un nombre suffisant d’objets, ceux-ci sont enlevés (cela peut éventuellement impliquer un déplacement des cases si, en enlevant les objets, on a vidé complètement une case) et on affiche le message « Objets retirés ».
* Si ce n’est pas le cas, l’inventaire est inchangé et on affiche le message « Inventaire insuffisant ».

## Étape 10 : proposer un menu

Le programme principal propose quatre choix à l’utilisateur :

* A pour ajouter des objets à l’inventaire ;
* R pour retirer des objets de l’inventaire ;
* I pour afficher l’inventaire (et le poids total) ;
* Q pour quitter le programme.

Dans les deux premiers cas, on demande tout d’abord à l’utilisateur de choisir les objets en question (via la fonction de l’étape 6) puis on les ajoute/retire. Si l’utilisateur rentre « Q », le programme se termine ; dans les autres cas, après avoir accompli la tâche demandée, le programme affiche à nouveau le menu.

## Étape 11 : rassembler deux inventaires

L’objectif est de faire en sorte d’ajouter le contenu d’un inventaire à un autre inventaire.

En effet, imaginez que le personnage meurt, son sac à dos est au sol et contient les objets qu’il avait avant sa mort. Lorsqu’il revient sur place, il veut récupérer les objets de son ancien inventaire. Il faut donc faire en sorte de pouvoir rassembler son ancien inventaire avec son nouvel inventaire qui est potentiellement (partiellement) rempli.

Écrivez une fonction qui permet de rassembler deux inventaires reçus en paramètres. Vous disposez déjà des fonctions nécessaires à la gestion de cette étape. Pensez à les utiliser à bon escient !

N’oubliez pas d’ajouter un nouveau choix au menu à la fonction principale et de faire appel à cette fonction.

## Étape 12 : consolider un inventaire

L’objectif est de regrouper les différentes piles d’un même type d’objets.

Considérez que vous avez déjà dans votre inventaire les objets suivants : 1 arc, 7 torches, 1 épée courte, 32 flèches, 25 torches. On désire faire en sorte que les deux piles de 7 et 25 torches soient réunies en une seule pile de 32 torches.

Si plusieurs types d’objets se retrouvent dans cette situation, le but est de regrouper les piles afin de n’avoir qu’une case par type d’objets. Le fait de regrouper des données similaires s’appelle la consolidation.

Écrivez une fonction qui permet de consolider un inventaire reçu en paramètre.

N’oubliez pas d’ajouter un nouveau choix au menu à la fonction principale et de faire appel à cette fonction.

## [Dépassement] Étape 13 : ajouter une taille maximum pour les piles d’objets

Recopiez votre code dans un nouveau projet afin de ne pas perdre votre première version. C’est le code de ce nouveau projet que vous allez modifier.

Ajoutez au bon endroit, dans vos structures, le fait qu’à chaque type d’objet est aussi associé une taille maximum de pile d’objets.

Adaptez ensuite l’ensemble du programme pour respecter cette nouvelle limite.