TP SUR LA GESTION DYNAMIQUE DE LA MÉMOIRE

Création et gestion de playlists musicales

On souhaite réaliser un programme permettant de créer et de gérer des playlists de morceaux de musique.
Un morceau est défini par son titre, le nom de l'artiste ou du groupe, son année d'enregistrement et sa durée.

Une playlist porte un nom et contient une liste de morceaux.

Un morceau de musique n'est défini qu'une seule fois mais peut être référencé dans plusieurs playlists.

Le programme devra proposer un menu permettant par exemple de :

- Créer une nouvelle playlist
- Charger une playList à partir d'un fichier texte
- Afficher une playList
- Ajouter un nouveau morceau dans une playList
- Ajouter un morceau d'une playList dans une autre playList
- Rechercher un morceau
- Enlever un morceau d'une playList
- Supprimer une playlist
- Sauvegarder une playList dans un fichier texte

…

I. Réflexion et conception

1. Modélisation des morceaux et des playlists

- **Pourquoi n'est-il pas souhaitable de dupliquer les informations d'un même morceau dans différentes playlists ? Citer au moins deux inconvénients d'une telle duplication.**

On envisage deux solutions :

- Solution A : Chaque playlist contient un tableau de structures *Morceau*.
- Solution B : Chaque playlist contient un tableau de pointeurs vers des structures *Morceau*.

- **Faire les schémas mémoire correspondant à chaque solution.**
- **Les deux solutions envisagées permettent-elles le partage d'un même morceau par plusieurs playlists ? Que se passerait-il si un morceau était modifié dans une des playlists ? Pourquoi la solution B est-elle mieux adaptée ?**

Pour la suite, nous allons donc adopter la solution B :

- Les morceaux de musique seront stockés dans des structures allouées dynamiquement.
- Une playlist contiendra un tableau dynamique de pointeurs sur des structures Morceau.
- Les différentes playlists seront stockées dans un tableau dynamique de structures PlayList.

- **Que se passe-t-il si on enlève un morceau d'une playlist ? Comment faire si on souhaite supprimer complètement un morceau (l'enlever de toutes les playlists dans lesquelles il apparait) ? Quand on enlève un morceau d'une playlist, comment savoir s'il faut libérer la structure Morceau ?**

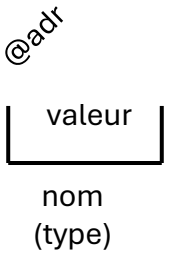
En vous aidant des réflexions précédentes, lister les informations qu'il faut stocker sur les morceaux et sur les playlists pour pouvoir tout gérer dynamiquement correctement.

2. Fonctionnement et représentations mémoire

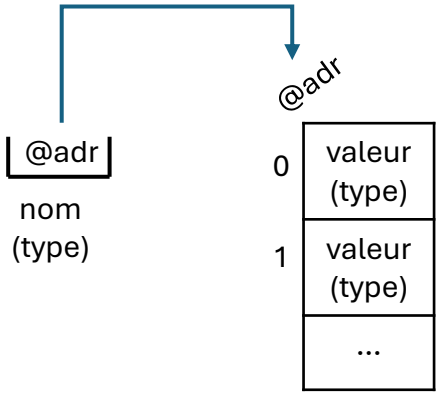
a. Schémas mémoire : Notations et formalisme

Pour réaliser les schémas mémoires demandés, respectez le formalisme et les notations des exemples ci-dessous. Précisez les identificateurs (noms de variables, noms de champs, types), indiquez des adresses mémoire, différenciez les emplacements stockés dans la pile de ceux stockés dans le tas.

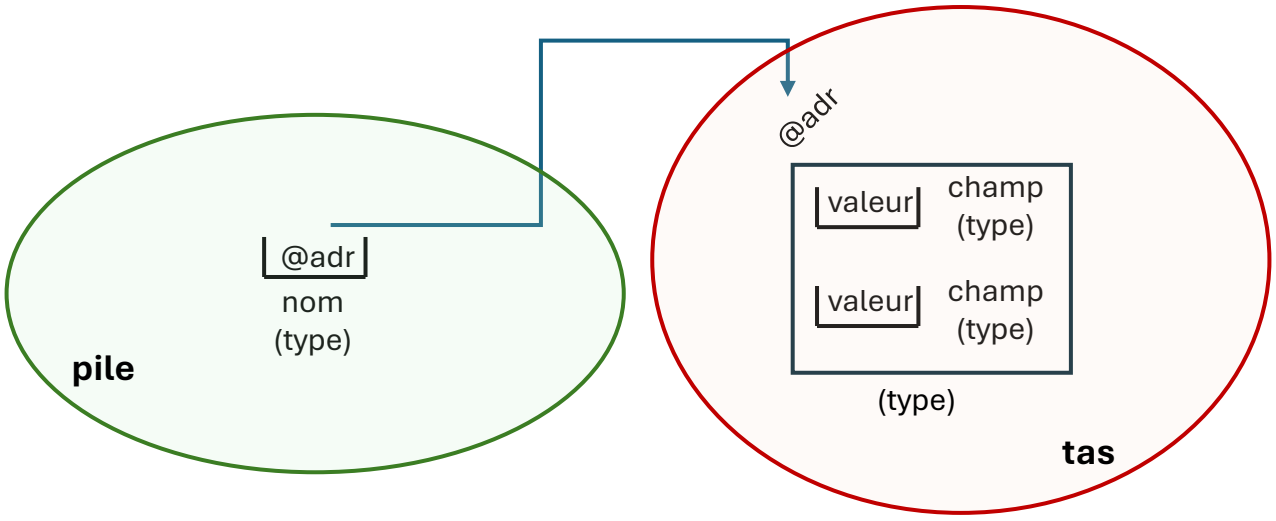
- Représentation d'une variable de type simple :



- Représentation d'un tableau à une dimension :



- Représentation d'un pointeur automatique sur structure dynamique :



a. Création de playlists

On crée un tableau dynamique de 2 playlists :

- *Top70*
- *MesFavoris*

A la création, on prévoit que les playlists puissent contenir 3 morceaux (qui seront ajoutés plus tard).

Dessiner la mémoire après la création du tableau

b. Création, ajout et partage de morceaux dans les playlists.

On crée dynamiquement trois morceaux :

- *Yesterday* – The Beatles – 1965 - 125 secondes
- *London Calling* – The Clash – 1979 - 200 secondes
- *Wish You Were here* – Pink Floyd – 1975 – 334 secondes

On ajoute :

- *Les 3 morceaux dans MesFavoris*
- *London Calling et Wish You Were here* dans *Top70*

Mettre à jour le schéma mémoire

Montrer clairement que :

Certaines cases des tableaux sont inutilisées.

Certains morceaux sont pointés par deux playlists.

c. Suppression d'un morceau d'une playlist

Que faut-il faire si :

• On supprime *Yesterday* de *MesFavoris ?*

• On supprime *London Calling* de *MesFavoris ?*

Mettre à jour le schéma mémoire

d. Suppression d’une playlist

On supprime la playlist *Top70*. **Que doit-on libérer exactement ? Que ne doit-on pas libérer ?**

e. Ajout après suppression

On souhaite maintenant créer un nouveau morceau :

Smells Like Teen Spirit - Nirvana - 1991 – 301 secondes

et l’ajouter à la playList *MesFavoris*.

Comment faire ? Quelle case utilisée ?

Mettre à jour le schéma mémoire

f. Redimensionnement des playlists

Quand faut-il redimensionner (agrandir ou diminuer) les playlists ? Comment procéder ?

On souhaite augmenter la taille de la playlist *MesFavoris* pour qu’elle puisse accueillir 5 morceaux.

Mettre à jour le schéma mémoire

g. Fin du programme

Que faut-il libérer et dans quel ordre ?

3. Identification des données

En complétant le tableau ci-dessous, récapituler toutes les informations à stocker et sous quelles formes.

D'après nos réflexions précédentes :

- Chaque morceau contiendra un champ *nbReferences* indiquant le nombre de playlists dans lesquelles il apparaît.
  Ce champ permet de savoir si un morceau est encore utilisé ou s'il peut être libéré.
- Une playlist contiendra deux champs *capacité* et *nbMorceaux*. Le champ *capacite* indique la taille du tableau alloué, tandis que *nbMorceaux* indique le nombre réel de morceaux présents dans la playlist (nombre de cases utilisées du tableau).

Données	Types
Morceau • titre …	type structuré (t_morceau) • tableau dynamique de caractères (char*) …
PlayList …	type structuré
…	…
Tableau dynamique de playlists	..
Nombre de playlists	

II. Découpage modulaire et identification des principaux traitements

Nous proposons le découpage modulaire suivant :

- Module morceau
- Module playList
- Module ihm

La couche métier est la partie du programme qui gère les données et qui contient les algorithmes de calculs et de résolution des problèmes, indépendamment de l'utilisateur. Elle contient les modules morceau et playlist. La couche métier ne doit fait appel à aucune fonction de saisie ou d'affichage.

Pour chaque module, lister les principaux traitements en indiquant les données sur lesquels ils portent (entrées) et les résultats en sortie.

Modules	Traitements	Entrées	Sorties
morceau	Créer et initialiser un morceau	Le titre, l'interprète, l'année et la durée	L'adresse de la structure allouée et remplie.
	Détruire un morceau.	L'adresse du morceau à libérer	
	Incrémenter le nombre de références	Le nombre de références (par adresse car modifié)	
	...		
playlist	Créer le tableau de playlists	Le nombre de playlists	L'adresse du tableau alloué.
	Initialiser une playlist		
	Ajouter un morceau à une playlist		
...			

III.Implémentation et tests

Implémenter pas à pas les différents modules.

Chaque module est composé d'un fichier header .h et d'un fichier source .c.

Les fichiers header contiennent les directives de pré-compilation (aucune instruction exécutable) :

- Inclusion des headers des bibliothèques ou autres modules utilisés
- Définition des constantes symboliques
- Définition des types structurés
- Prototypes commentés des sous-programmes

Les fichiers sources contiennent :

- Inclusion des headers des bibliothèques ou modules utilisés
- Implémentation des sous-programmes

Utiliser un *main()* temporaire pour tester vos sous-programmes au fur et à mesure.

Commencer par coder les sous-programmes de saisies et d'affichage pour pouvoir tester.

Rappels :

- **Tout est géré dynamiquement (les chaines de caractères, les morceaux, les playlists …)**
- Les allocations (malloc , realloc) doivent être suivi d'un test de validité (vérifier si le pointeur est NULL).