



# **INF1220 : STRUCTURES DE DONNÉES ET PROGRAMMATION C**

AMOUZOU Grâce Dorcas Akpéné / EPL / 2024-2025 M

---

# SÉANCE 1 - PARTIE I : CONTRAT PÉDAGOGIQUE



# OBJECTIFS

- Identifier les différents types de structures de données ;
- Utiliser les structures de données dans des programmes ;
- Décrire les structures de données les plus courantes ;
- Sélectionner la structure de donnée adaptée à un algorithme donné.



# PRÉREQUIS

- Algorithmique
- Langage C

---

# SÉANCE 1 - PARTIE II : INTRODUCTION



## Définitions des SDD

C'est une structure logique destinée à contenir des données afin de leur donner une organisation rationnelle destinée à organiser, traiter, extraire et stocker des données.

Une structure de données est un format spécial destiné à organiser, traiter, extraire et stocker des données. S'il existe plusieurs types de structures plus ou moins complexes, tous visent à organiser les données pour répondre à un besoin précis, afin de pouvoir y accéder et les traiter de façon appropriée.



## Importance des SDD 1/2

Les structures de données sont indispensables pour gérer efficacement de grandes quantités de données, comme les informations stockées dans une base de données ou des services d'indexation.

La bonne gestion d'un système de données exige la capacité d'identifier la mémoire allouée, les relations entre les données et les processus de données. Or, les structures de données facilitent ces opérations.



## Importance des SDD 2/2

Par ailleurs, non seulement il est important d'utiliser des structures de données, mais il est également indispensable de choisir la structure adaptée à chaque tâche.

Choisir la mauvaise structure de données pourrait entraîner un ralentissement des temps de traitement ou une absence de réponse du code.



---

# Types de SDD



## TYPE 1

"Imagine l'équipe de Pokémon d'un dresseur : il peut accéder directement au 1er, 3e ou 6e Pokémon sans passer par les autres. Ils sont bien rangés dans leur Pokéball, dans un ordre fixe."

"Imagine les lits des 7 nains, tous alignés. Tu peux aller directement voir le lit de Grincheux ou Joyeux sans devoir réveiller ceux avant."

"Imagine un plateau de fruits avec des cases numérotées. Tu veux le fruit n°4 ? Tu peux le prendre tout de suite, sans toucher aux autres."



## TYPE 1 : Tableau

Un tableau stocke un ensemble d'éléments dans des emplacements de mémoire contigus. Les éléments de même type sont stockés ensemble afin de faciliter le calcul de leur emplacement ou leur extraction. La longueur d'un tableau peut être fixe ou variable.

<b>Valeur</b>	<b>45</b>	<b>154</b>	<b>58</b>	<b>78</b>	<b>31</b>	<b>5</b>	<b>74</b>
<b>Index</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>



## TYPE 2

"Imagine Luffy partant à l'aventure, et chaque île qu'il visite est connectée à la suivante par un log pose. Il ne peut pas sauter directement à la fin : il doit suivre la route, île par île, dans l'ordre. Chaque île connaît la suivante, mais pas celle d'avant."

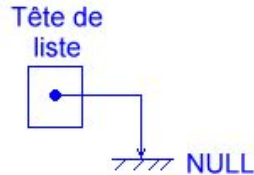
"Imagine qu'on cherche Papa, et qu'on interroge chaque membre de la famille à la suite. Chacun nous dit où chercher ensuite. On ne peut pas aller directement à Papa, on doit suivre la piste, un par un."

"Chaque rêve contient un autre rêve à l'intérieur, et pour remonter à la réalité, il faut passer par chaque niveau dans l'ordre. On ne peut pas sauter directement de la fin au début."

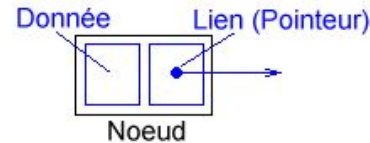
## TYPE 2 : Liste chaînée

Une liste chaînée stocke un ensemble d'éléments de façon linéaire. Chaque élément ou nœud d'une liste chaînée contient un élément de données ainsi qu'une référence, ou lien, vers l'élément suivant de la liste.

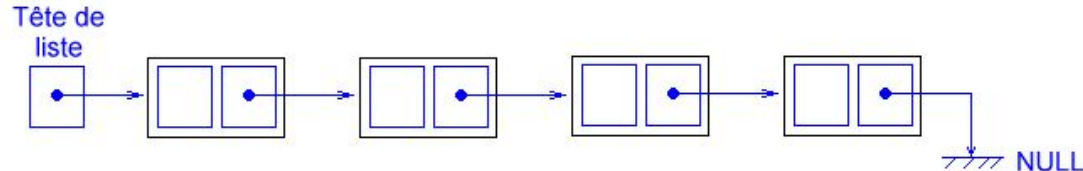
Liste vide :



Légende :



Exemple de liste non vide :





## TYPE 3

"Imagine Batman empilant ses dossiers de mission. Il prend toujours celui du dessus. S'il veut revoir un ancien dossier, il doit d'abord enlever ceux au-dessus."

"Imagine une pile de livres. Tu peux seulement lire le livre du haut. Si tu veux lire celui du bas, il faut tous les enlever un par un."

"Imagine que tu fais la vaisselle, volontairement bien sûr, et que ton frère en rajoute une assiette. Il faut laver celle-là avant de laver les autres."

## TYPE 3 : Pile

Une pile stocke un ensemble d'éléments en suivant l'ordre linéaire dans lequel les opérations sont appliquées. Par exemple, dernier entré, premier sorti (LIFO, Last In First Out) ou premier entré, dernier sorti (FILO, First In Last Out).





## TYPE 4

"Imagine les élèves de Poudlard en file devant la Grande Salle. Le premier arrivé est le premier à entrer, les autres attendent leur tour."

"Imagine des gens qui montent dans un bus. Celui arrivé en premier monte en premier, les autres suivent dans l'ordre."

"Chez ayimolouto, le vendeur ou la vendeuse sert les clients un à un. Les commandes sont servies dans l'ordre d'arrivée."



## TYPE 4 : File

Une file stocke un ensemble d'éléments de façon similaire à une pile, mais l'ordre des opérations ne peut être que de type premier entré, premier sorti (FIFO First In First Out) ou dernier entré, dernier sorti (LILO Last In Last Out).





## TYPE 5

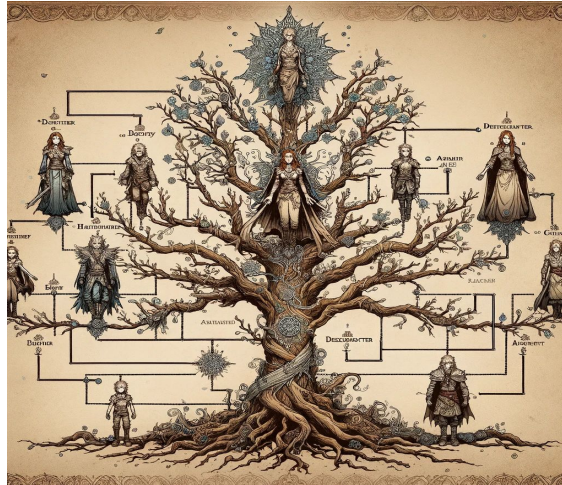
"Imagine l'arbre généalogique d'Elrond. Il est en haut, puis ses enfants en dessous, puis les enfants de ses enfants. Chaque niveau descend de l'autre."

"Imagine l'arbre généalogique des Targaryen : Aegon au sommet, puis ses descendants, chacun connecté à ses enfants."

"Imagine Lucious Lyon tout en haut, puis ses responsables, puis les employés. Chacun dépend de quelqu'un au-dessus."

## TYPE 5 : Arbre

Un arbre stocke un ensemble d'éléments sous une forme hiérarchique abstraite. Chaque nœud est relié aux autres et peut contenir plusieurs sous-valeurs appelées enfants.





## TYPE 6

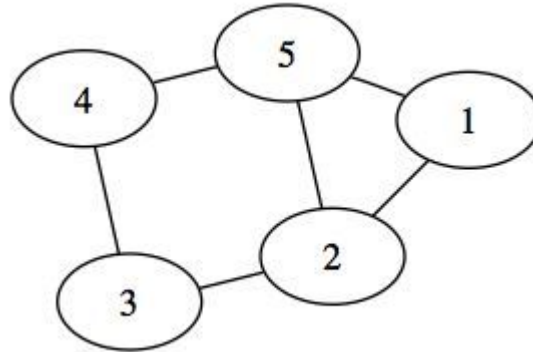
"Imagine chaque univers du multivers comme un point, et les portails entre eux comme des connexions. Spider-Man peut passer de n'importe quel monde à un autre selon les connexions, pas forcément dans un ordre précis."


"Imagine un plan de métro avec des stations connectées entre elles. Certaines lignes se croisent, certaines stations ont plusieurs chemins pour y accéder."

"Sur Tiktok / Instagram, chaque personne est un noeud et chaque lien représente un "follow".

## TYPE 6 : Graphe

Un graphe stocke un ensemble d'éléments de façon non linéaire. Il se compose d'un ensemble fini de nœuds, appelés sommets, et de lignes, les arêtes, qui relient les sommets entre eux. Les graphes permettent notamment de représenter des systèmes réels, comme des réseaux informatiques.





Dans les langages de programmation, les structures de données permettent d'organiser le code et les informations dans l'espace numérique. Par exemple, les listes et les dictionnaires Python, ou les tableaux et objets JavaScript sont des structures de codage couramment utilisées pour stocker et récupérer des informations. Les structures de données constituent également un élément essentiel dans la conception de logiciels efficaces.

---

# SÉANCE 1 - PARTIE III : LES LISTES CHAÎNÉES



## D'abord, les TABLEAUX

Un tableau est une structure de données représentant une séquence finie d'éléments auxquels on peut accéder efficacement par leur position, ou indice, dans la séquence.

Les tableaux vérifient généralement les propriétés suivantes :

- tous les éléments ont le même type de base ;
- le nombre d'éléments stockés est fixé ;
- l'accès et la modification de l'élément numéro  $i$  est indépendant du nombre d'éléments dans le tableau.





## DÉFINITION des LC

Une liste chaînée est une structure de donnée permettant, comme les tableaux, de stocker plusieurs valeurs de même type, mais qui soit de taille variable, contrairement aux tableaux.

On la nomme ainsi car elle est similaire à une chaîne composée de maillon : chaque maillon est relié au maillon suivant, de sorte qu'en ne tenant que le premier maillon, on peut accéder (en « déroulant » la chaîne) à tous les maillons de la chaîne. Par ailleurs, on peut facilement agrandir ou rétrécir la chaîne en lui ajoutant ou retirant des maillons.



## Caractéristiques des LC

Les éléments successifs sont reliés par des pointeurs ;

Le dernier élément indique NULL ;

La taille peut augmenter pendant l'exécution d'un programme ;

Peut être rallongée aussi longtemps que nécessaire (jusqu'à épuisement de la mémoire des systèmes)

Ne gaspille pas d'espace mémoire (mais prend un peu de mémoire supplémentaire pour les pointeurs). La mémoire est allouée au fur et à mesure que la liste s'allonge.



## Comparaison entre Tableaux et LC $\frac{1}{2}$ : Les Tableaux

Simple et facile à utiliser

Accès plus rapide aux éléments (accès permanent)

Préallocation de toute la mémoire nécessaire au départ et gaspillage d'espace mémoire pour les indexes dans le tableau qui sont vides.

Taille fixe : La taille du tableau est statique (préciser la taille du tableau avant de l'utiliser).

Insertion complexe basée sur la position : Pour insérer un élément à une position donnée, on peut avoir besoin de déplacer les éléments existants.



## Comparaison entre Tableaux et LC 2/2 : Les LC

Taille ajustable (allocation dynamique du stockage)

Espace mémoire mieux géré

Temps d'accès aux données potentiellement élevé

Bien que l'allocation dynamique du stockage soit un grand avantage, les frais généraux liés au stockage et à l'extraction de données peuvent faire une grande différence.

Gaspillage de la mémoire en termes de points de référence supplémentaires.



## Opérations sur les LC

- Insertion ;
- Suppression ;
- Recherche ;
- La liste est-elle vide ?
- La liste est-elle pleine ?
- Concaténation ;
- Tri ;
- Etc.

**SÉANCE 1 :**  
**END**

—