



STRUCTURES DE DONNÉES

AMOUZOU Grâce
gamouzou@univ-lome.tg
92 11 31 05
2023 - 2024



Chapitre 0



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.



Caractéristiques des SDD

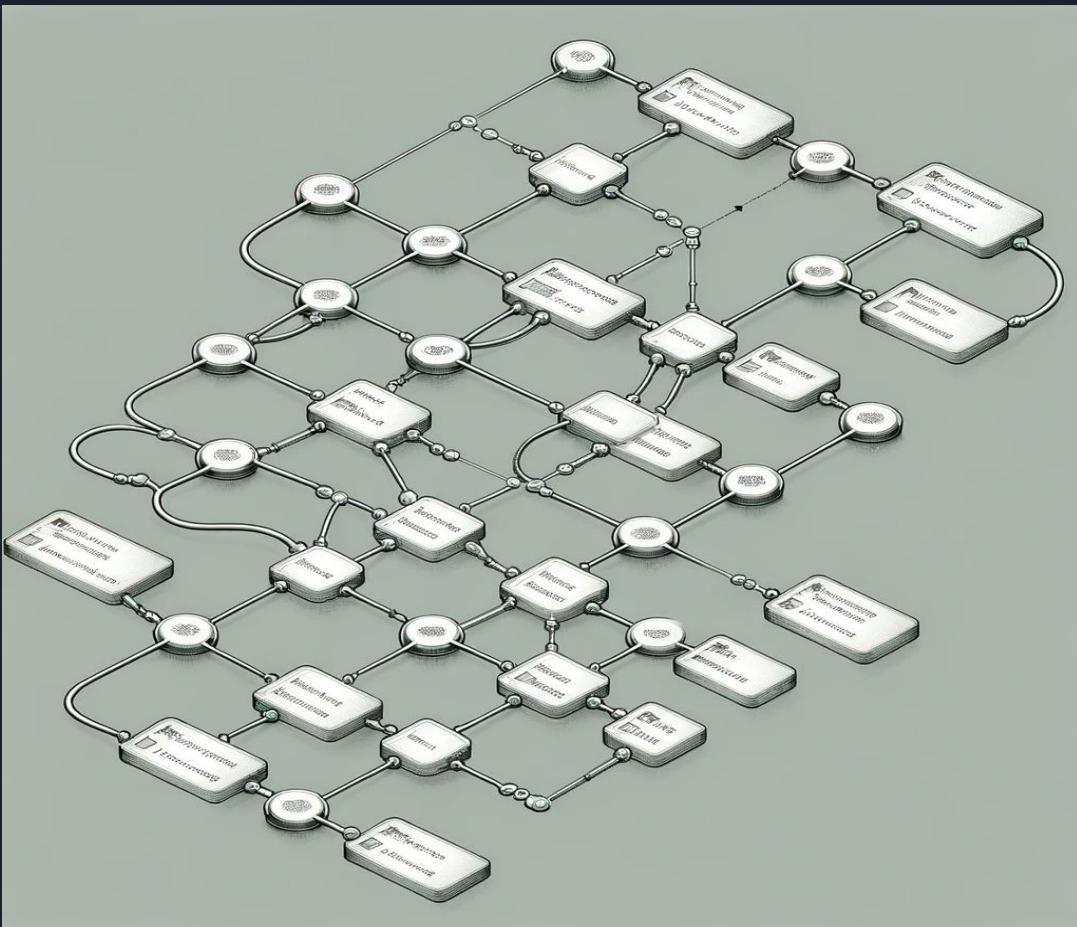
- 
- 01 **Linéaires ou non linéaires** : indique si les éléments de données sont organisés chronologiquement, comme dans un tableau, ou de façon non ordonnée, comme dans un graphe.
 - 02 **Homogènes ou non homogènes** : indique si tous les éléments de données d'un référentiel spécifique sont du même type ou de types différents.
 - 03 **Statiques ou dynamiques** : décrit la façon dont les structures de données sont compilées. Les structures statiques présentent des tailles, des structures et des emplacements de mémoire fixes au moment de la compilation. Dans une structure de données dynamique, en revanche, la taille, les structures et les emplacements de mémoire peuvent diminuer ou s'agrandir, selon l'utilisation.

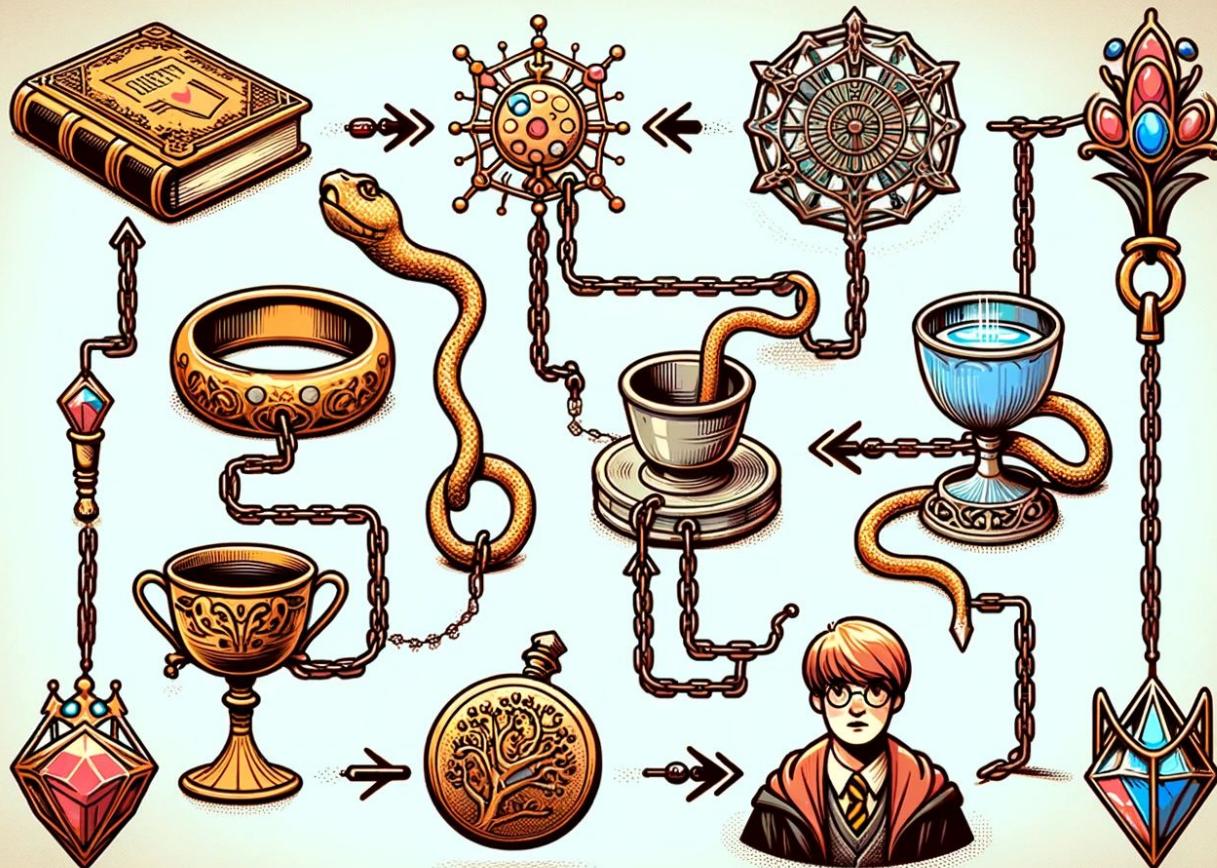


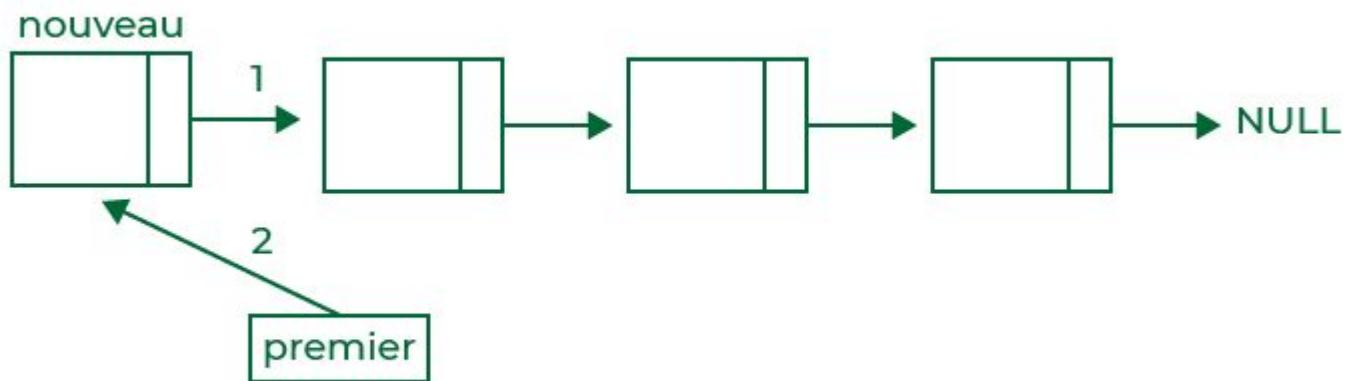
Type de SDD 1/7

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.









Type de SDD 2/7

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.



	0	1	2
0	20	16	14
1	17	21	15

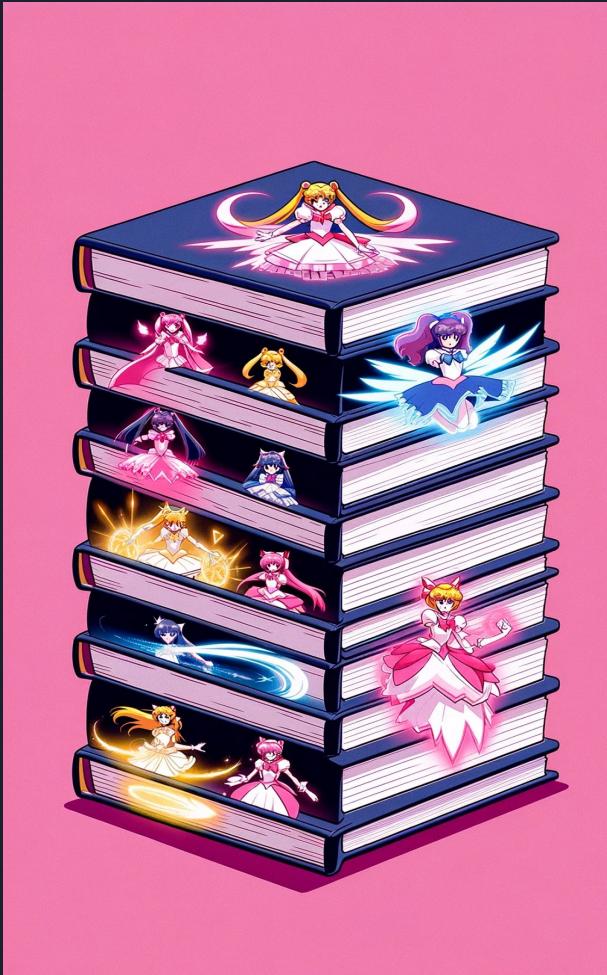
un dimension deux dimensions

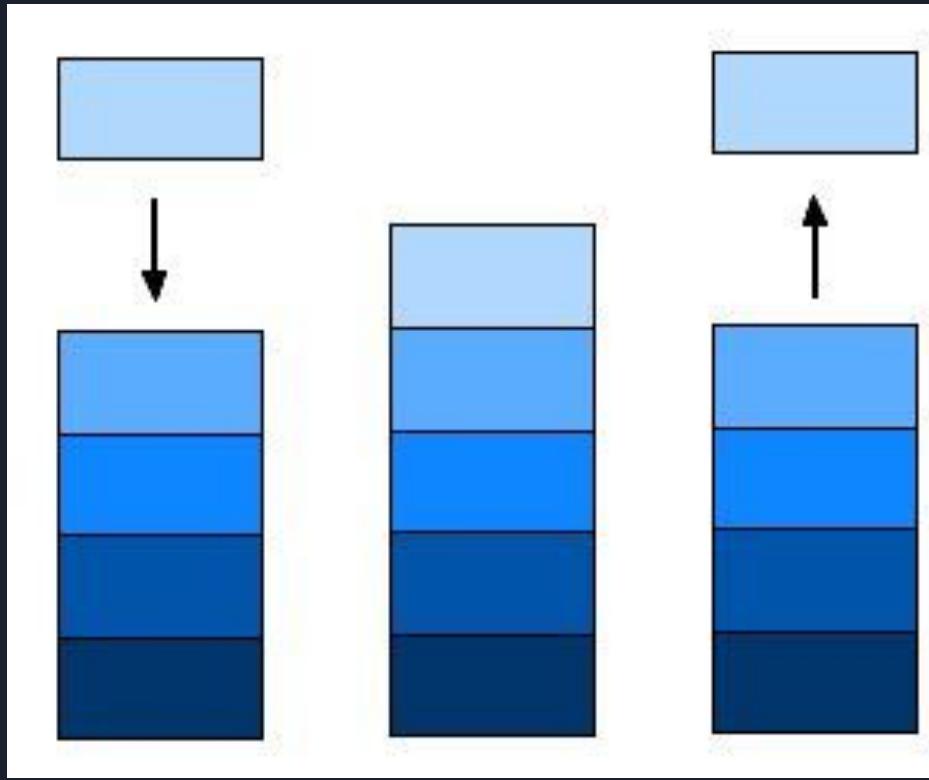


Type de SDD 3/7

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, premier sorti (FIFO, First In First Out).





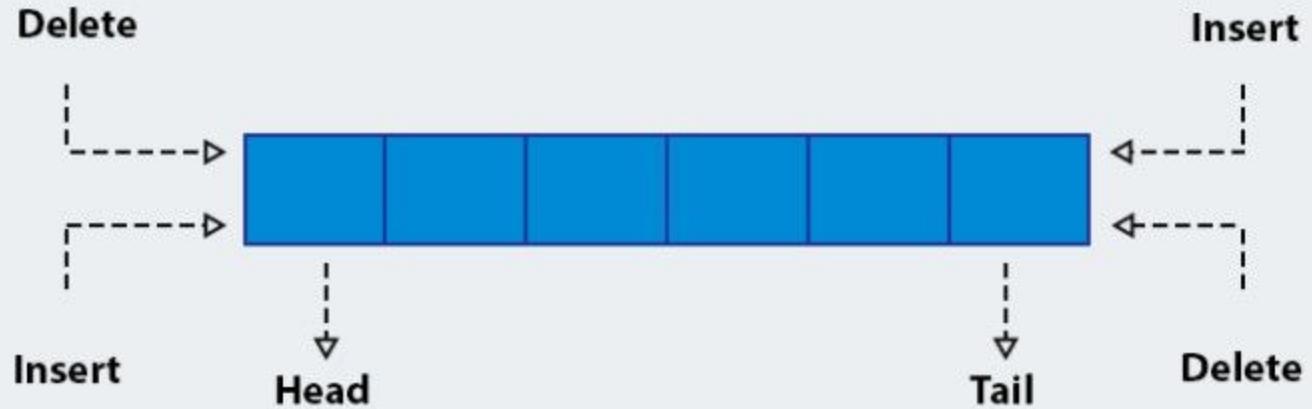


Type de SDD 4/7

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.



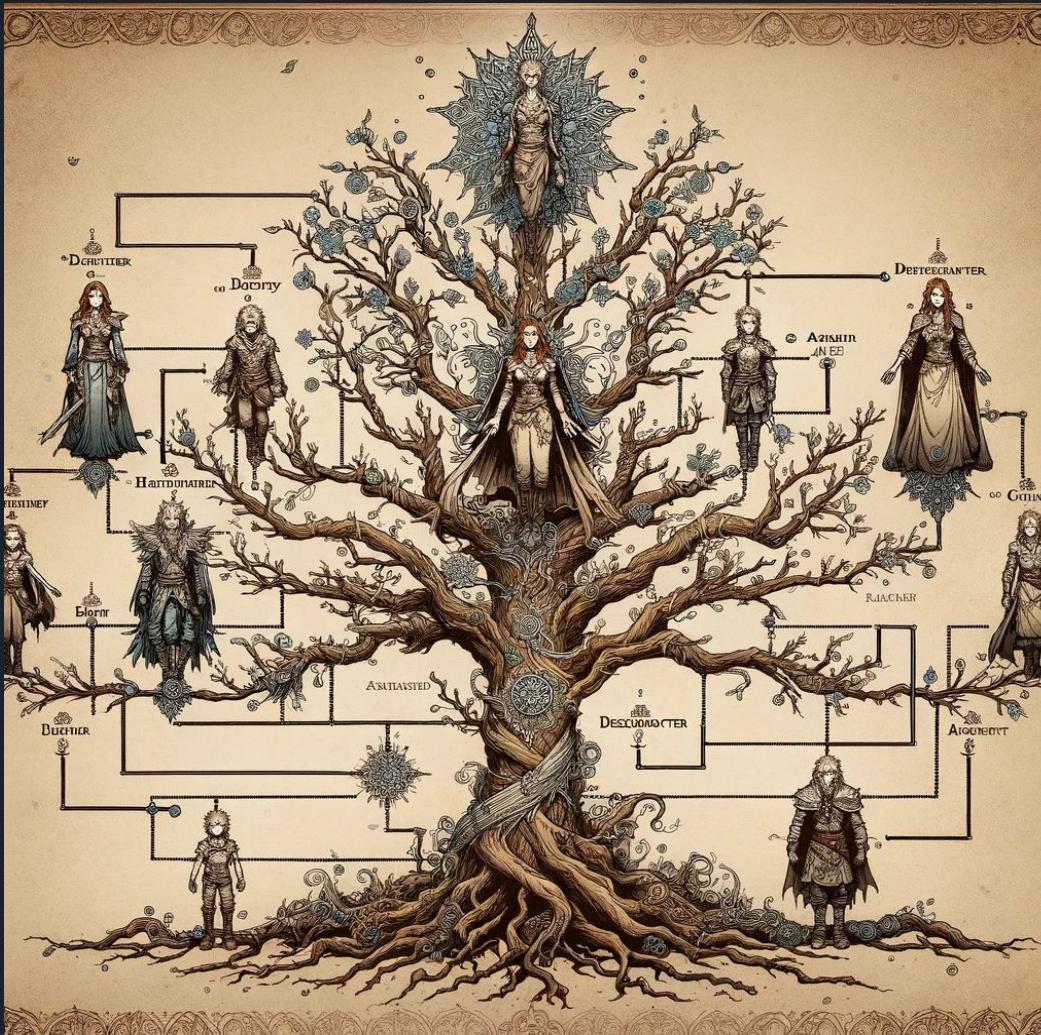


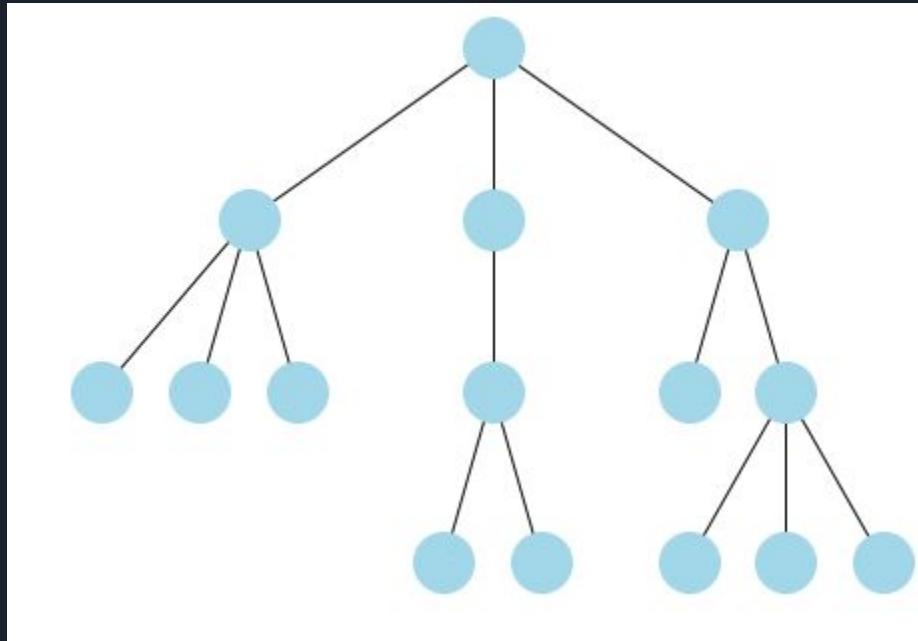


Type de SDD 5/7

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 de SDD 6/7

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.



Type de SDD 7/7

Table de hachage.

Une table de hachage stocke un ensemble d'éléments dans un tableau associatif qui fait correspondre des clés à des valeurs. Cette structure utilise une fonction de hachage pour convertir un indice en tableau dont les alvéoles contiennent l'élément de données souhaité.



En règle générale, les structures de données servent à mettre en œuvre les formes physiques de types de données abstraits. Cela peut donner lieu à une multitude d'applications, comme l'affichage d'une base de données relationnelle sous forme d'arbre binaire.

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.



LES LISTES CHAÎNÉES



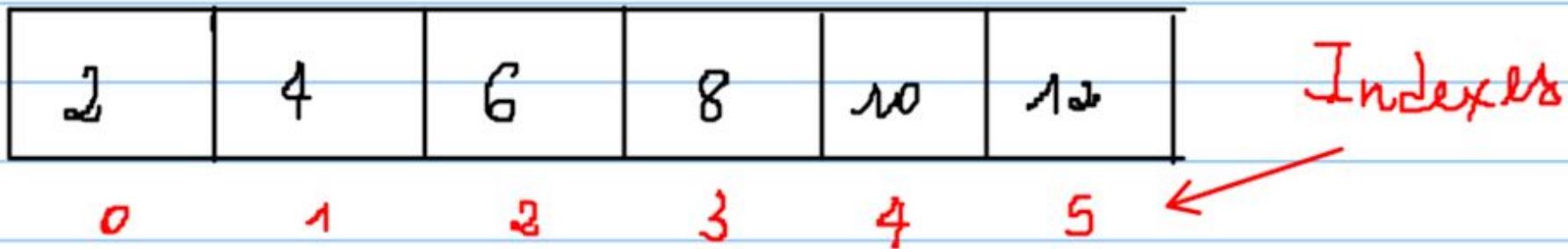
Tableaux

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 en temps constant $O(1)$, indépendant de i et du nombre d'éléments dans le tableau.

Exemple





Définitions 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 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.

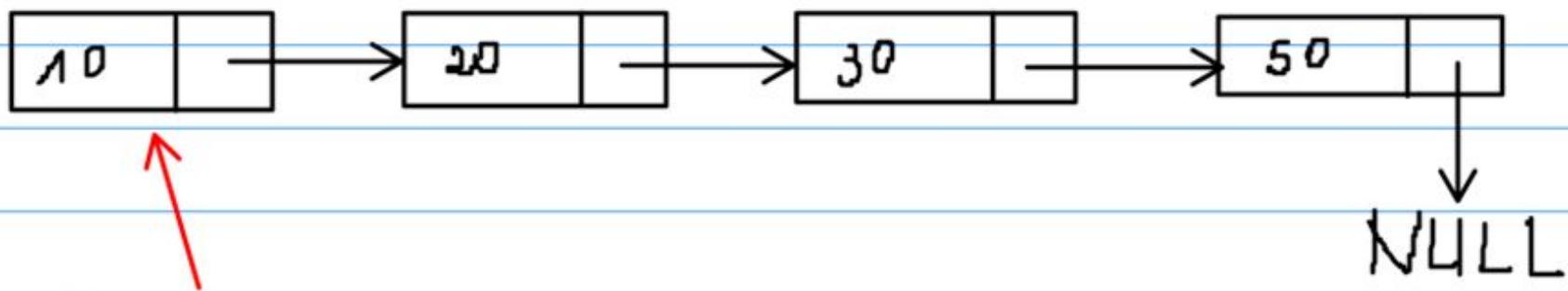


Caractéristiques des LC

Une liste chaînée possède les caractéristiques suivantes.

- 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.

Exemple



Premier



Comparaison entre Tableaux et LC 1/2

- Simple et facile à utiliser
- Accès plus rapide aux éléments (accès permanent)

INCONVENIENTS DES TABLEAUX

- Préallocation de toute la mémoire nécessaire au départ et gaspillage d'espace mémoire pour les indexées 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. Cela nous permettra d'insérer le nouvel élément à l'endroit souhaité. Si la position à laquelle nous voulons ajouter un élément est au début (en première position), alors, l'opération de déplacement est plus coûteuse (en temps d'exécution, vu que, évidemment, il faudra déplacer tous les éléments du tableau).



Comparaison entre Tableaux et LC 2/2

- Taille ajustable (allocation dynamique du stockage)
- Espace mémoire mieux géré

INCONVENIENTS DES LC

- 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. Parfois, les listes chaînées sont difficiles à manipuler. Si le dernier élément est supprimé, l'avant-dernier doit alors faire changer son pointeur pour contenir une référence sur NULL. Pour cela, il faut parcourir la liste pour trouver l'avant-dernier élément pour le faire pointer sur NULL.
- 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.

Fin du chapitre 0

