# **Concepts des langages**

## **TP**

Table des matières

[Concepts des langages 1](#_Toc193367642)

[**TP** 1](#_Toc193367643)

[Exercice n°2 : 2](#_Toc193367644)

[Exercice n°3 : 2](#_Toc193367645)

### Exercice n°2 :

Je choisis de représenter ma pile par une liste simplement chaînée premièrement car une liste chaînée possède une taille dynamique, ce qui n’est pas le cas des tableaux statiques. Les tableaux dynamiques, eux ont une taille allouée à l’exécution, ce qui est intéressant, mais, ici, nous voulons pouvoir ajouter et supprimer des éléments dans notre pile et donc, avoir une taille dynamique. Si nous choisissons un tableau dynamique nous devrons gérer la taille et la mémoire à l’aide de realloc(). Ensuite, étant donné que chaque élément contient une valeur et un pointeur vers l’élément suivant l’accès aux données sera en O(n) ce qui est plus coûteux. Mais l’ajout et la suppression sera en O(1). Ici, nous préfèrerons avoir un ajout rapide et un accès coûteux, car il est très rare que nous ayons besoin de parcourir la pile entière, et les interactions avec une pile se font toujours avec le sommet de celle-ci.

### Exercice n°3 :

Je choisis de représenter ce tableau d’horaires de trains par un tableau dynamique de structures « horaire ». En effet, le fait que le tableau soit dynamique permet de pouvoir modifier la taille de celui-ci et donc d’ajouter et de supprimer des horaires à notre convenance grâce à l’allocation dynamique de la mémoire et les fonctions malloc(), realloc() et free(). L’accès aux données sera simple car notre tableau est à une seule dimension. De plus, le fait de manipuler les structures regroupe les différentes données de chaque horaire. On peut également ajouter que notre code sera plus lisible.